Secure Web Services für ambiente eingebettete Systeme

Dissertation
zur
Erlangung des akademischen Grades
Doktor-Ingenieur (Dr.-Ing.)
der Fakultät für Informatik und Elektrotechnik
der Universität Rostock

vorgelegt von
Sebastian Unger, geb. am 30.08.1983 in Wuppertal
aus Rostock

Rostock, 07. Oktober 2015

urn:nbn:de:gbv:28-diss2016-0024-0

Gutachter

Prof. Dr.-Ing. Dirk Timmermann
Universität Rostock
Fakultät für Informatik und Elektrotechnik
Institut für Angewandte Mikroelektronik und Datentechnik
Richard-Wagner-Straße 31, 18119 Rostock-Warnemünde

Prof. Dr. rer. nat. Stefan Fischer
Universität zu Lübeck
Institut für Telematik
Ratzeburger Allee 160, 23562 Lübeck

Prof. Dr. Heiko Krumm
Technische Universität Dortmund
Fakultät für Informatik
LS IV Rechnernetze und verteilte Systeme
Otto-Hahn-Straße 16, 44221 Dortmund

Datum der Verteidigung: 03.03.2016

Für Euch. Für uns.

Danksagungen

Bei Fertigstellung einer Arbeit wie der vorliegenden, ist es eine gute Tradition, dass der Autor in sich geht und reflektiert, ohne wen diese Arbeit so nicht zustande gekommen wäre. Ich schulde einigen Menschen tief empfundenen Dank, den ich hier ausdrücken möchte.

Schon zu Studienzeiten hat Prof. Timmermann meinen Werdegang maßgeblich beeinflusst. Mit seinen Vorlesungen begeisterte er mich für die Vertiefung der Technischen Informatik und aufgrund seiner Empfehlung durfte ich als Praktikant zur Toshiba Corporation nach Japan. Ihm habe ich ebenfalls zu verdanken, dass ich meine Diplomarbeit in Zusammenarbeit mit der BMW CarIT GmbH in München anfertigen durfte. So erhielt ich ebenfalls meinen Zugang zum interessanten Feld der Gerätesicherheit. Während meiner Promotion unterstützte er mich stets mit Rat und Tat und ermöglichte mir die Fertigstellung meiner Arbeit auch über meine Zeit als Stipendiat hinaus.

Auch meinem Zweitgutachter Prof. Cap gilt mein zutiefst empfundener Dank. Ohne seine Anregungen zu logischem Aufbau und argumentativen Ketten sowie Diskussionen zum Inhalt dieser Arbeit, hätte sie nicht diese Form erreicht.

Ich bedanke mich weiter bei allen Professoren des DFG Graduiertenkollegs 1424 „MuSAMA“ für interessante Diskussionen und Anregungen im Rahmen oder am Rande der zahlreichen Workshops. Auch meine Mitstipendiaten des Graduiertenkollegs gaben mir wertvolle Anregungen in den vielen Diskussionsgruppen. Meinen Kollegen am Institut für Angewandte Mikroelektronik und Datentechnik danke ich für beinahe fünf fantastische Jahre, für spannende Diskussionen und ein Verhältnis, das über „kollegial“ weit hinausgeht.

Schließlich danke ich meiner Familie, meiner Freundin Susanne und meinen Freunden für die jahrelange Unterstützung und Geduld.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verwendete Formelzeichen und Abkürzungen

6LoWPAN	IPv6 over low-power wireless personal area networks
AAL	Ambient Assisted Living
ABAC	Attribute-based Access Control
ACL	Access Control List
AH	Authentication Header
ASCII	American Standard Code for Information Interchange
BLE	Bluetooth Low Energy
CA	Certificate Authority
DLNA	Digital Living Network Alliance
DoS	Denial of Service
DPWS	Devices Profile for Web Services
DPWSec	Devices Profile for Web Services Security
DRM	Digital Rights Management
DTLS	Datagram Transport Layer Security
ECDH	Elliptic Curve Diffie Hellman
ESP	Encapsulated Security Payload

GPAP	General Purpose Access Point
HMAC	keyed-Hash Message Authentication Code
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
ICV	Integrity Check Value
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
ISO	International Organization for Standardization
IV	Initialisierungsvektor
KDF	Key Derivation Function
MAC	Message Authentication Code
MIC	Message Integrity Code
MITM	Man in the Middle
NAS	Network Attached Storage
NFC	Near Field Communication
NIST	National Institute of Standards and Technology
OOB	Out of Band
OSI	Open Systems Interconnection
PC	Personal Computer
PKI	Public Key Infrastructure
PLC	Power Line Communication
PoP	Proof of Possession
PRF	Pseudo Random Function

RBAC	Role-based Access Control
RC4	Ron's Code 4
RFC	Request for Comment
RTT	Round Trip Time
SAML	Security Assertion Markup Language
SCT	Security Context Token
SOA	Serviceorientierte Architekturen
SSL	Secure Sockets Layer
STiL	Service Technology-independant Language
TCP	Transmission Control Protocol
TLS	Transport Layer Security
UDP	User Datagram Protocol
UPnP	Universal Plug and Play
URN	Uniform Resource Name
UUID	Universal Unique Identifier
VPN	Virtual Private Network
WEP	Wireless Equivalent Privacy
WLAN	Wireless Local Area Network
WPA	Wireless Protected Access
WS	Web Services

Kapitel 1

Einleitung

Pervasive Computing, *intelligente Umgebungen*, *Internet der Dinge* – Visionen wie die von MARK WEISER in seinem vielbeachteten Aufsatz zu Beginn der neunziger Jahre [Wei91] werden real. *Smart Metering* und *Smart Grid* optimieren unseren Stromverbrauch, unser *Smart Home* überwacht unsere Beleuchtung und Belüftung und sorgt für einen Kühlschrank, der entsprechend unserer Ernährungsgewohnheiten gefüllt ist. In wenigen Jahren wird uns das *Ambient Assisted Living* ermöglichen, auch im hohen Alter in unserem eigenen Haus zu wohnen, denn digitale, unsichtbare Assistenten überwachen unbemerkt unseren Gesundheitszustand, unsere Medikamenteneinnahme und unsere Vitalparameter. Stößt uns doch etwas zu, ruft dieser unsichtbare Assistent menschliche Hilfe.

Diese Entwicklung geht mit dem nächsten großen Paradigmenwechsel des Computings einher. Nachdem das Erscheinen von PCs dazu führte, dass nicht mehr viele Nutzer einen Computer bedienten sondern nur noch einer, vollzieht sich dieser Tage der Wandel, dass ein Nutzer viele Computer bedient. Neben PC, Notebook, Tablet-PC, Smartphone und Spielekonsole beherbergen Smart-TV, Waschmaschine, Kaffeevollautomat, intelligente Beleuchtungskonzepte und Türschlösser ebenfalls Recheneinheiten. Man spricht von eingebetteten Systemen, da sie als Computer in andere „Dinge“ eingebettet sind, deren vordergründige Funktion kein Computer ist. Laut HOFFMAN et al handelte es sich bereits 2007 bei 98 % aller programmierbaren digitalen Geräte um eingebettete Systeme ([HBE+07]), STAMMBERGER et al. geben 90% für das Jahr 2010 an und erwarten 99% im Jahr 2020 ([SSAC10]).

Mit diesem Maß der Durchdringung all unserer Lebensbereiche steigt auch das Maß der Angreifbarkeit. All diese Geräte geben Informationen über uns preis, die unsere Privatsphäre vollständig auflösen, wenn man sie vor dem Hintergrund der Enthüllungen von Edward Snowden und dem Wissen um unsere vollständige Überwachung durch Geheimdienste betrachtet. Dass es sich dabei nicht nur um theoretische Möglichkeiten handelt, zeigen verschiedene Beispiele aus jüngster Vergangenheit. So gelang es dem bekannten Sicherheitsforscher Barnaby Jack 2012, durch Reverse-Engineering Zugriff auf die Funkschnittstelle von Herzschrittmachern zu erhalten und zeigte, dass es ihm möglich wäre, seinen Träger tödliche Stromschläge erleiden zu lassen ([Pau12]).

Während ein Artikel ([Fle15]) des deutschen Nachrichtenmagazins Technology Review im Jahr 2015 die Gefahr sieht, dass die Millionen eingebetteten Geräte durch die mangelnden Sicherheitsvorkehrungen mit wenig Aufwand gekapert und als Botnetze andere Infrastrukturen angreifen, ist das einem Unbekannten in seinem „Internet Census 2012“ längst gelungen [Unb13].

Lohnenswerte Ziele für Angriffe solcher Botnetze sind neben Industrie- und Versorgungsanlagen oder Regierungsstellen ebenfalls medizinische Einrichtungen. Unbemerkt und aus der Ferne ließen sich bei mangelnden Sicherheitsvorkehrungen belastende Patientendaten z.B. politischer Gegner sammeln. Vor dem Hintergrund der aktuell stattfindenden Vernetzung von medizinischen Geräten im Operationssaal ([KKU+15]), aktuellen Untersuchungen zum Stand der Sicherheit und erfolgreiche unerlaubte Fernzugriffe auf Infusionspumpen ([Wen15, EM14]), zeigt sich dringender Handlungsbedarf.

Die vorliegende Arbeit befasst sich mit der Sicherheit für verteilte, eingebettete Systeme als Bestandteil intelligenter Umgebungen. Im weiteren Verlauf dieses Kapitels wird diese Arbeit zunächst in das umfassende Feld der Sicherheit für verteilte eingebettete Systeme eingeordnet. Anschließend wird der Aufbau dieser Arbeit beschrieben. Abschließend erfolgt eine kurze Festlegung begrifflicher Konventionen.

1.1 Einordnung der Arbeit

Die Einordnung der vorliegenden Arbeit erfolgt anhand der Abbildung 1.1. Das Gebiet Sicherheit für eingebettete Systeme – wie auch für Computersys-

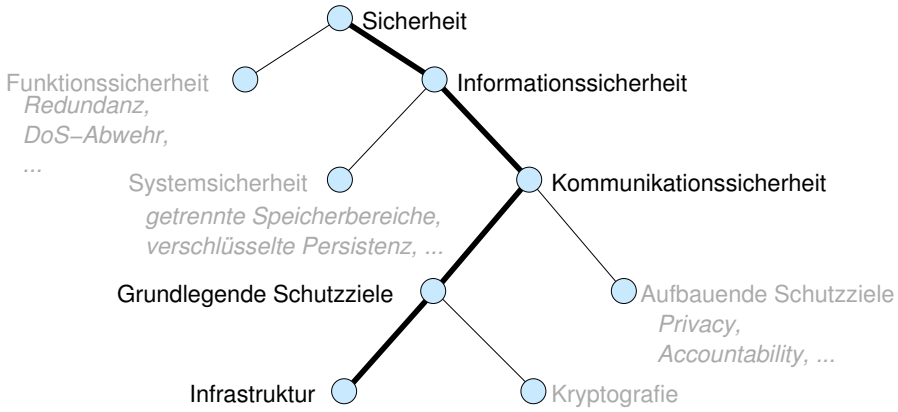


Abbildung 1.1: Einordnung der vorliegenden Arbeit

teme allgemein – muss zunächst in die Bereiche der *Funktionssicherheit* (im Englischen meist Safety) und der *Informationssicherheit* (englisch Security) unterteilt werden ([Eck08]). Unter dem Begriff Funktionssicherheit werden alle Maßnahmen zusammengefasst, die einen kontinuierlichen, unterbrechungsfreien und ungestörten Arbeitsfluss eines Computers oder eines Netzwerks begünstigen, wie z.B. unterbrechungsfreie Stromversorgungen gegen Stromausfall, redundante Steuerungssysteme gegen Ausfall einzelner Komponenten oder Firewalls zur Abwehr von Denial-of-Service-Angriffen.

Ziel der Informationssicherheit ist es, jegliche Daten zu schützen. Das bedeutet, zu gewährleisten, dass alle Daten, die gespeichert, verarbeitet oder übertragen werden, aus zuverlässiger Quelle stammen und vor unberechtigtem Zugriff geschützt werden. Mit unterschiedlichen Mitteln werden hier Bedrohungen adressiert, die im eigenen System lauern (*Systemsicherheit*) oder außerhalb, z.B. während der Datenübertragung (*Kommunikationssicherheit*). Die Systemsicherheit wird meist durch Mittel des Betriebssystems hergestellt wie z.B. die Gewährleistung getrennter Speicherbereiche für verschiedene Prozesse.

Zur Wahrung der Kommunikationssicherheit existieren zunächst die vier *grundlegenden Schutzziele* Authentizität, Integrität, Vertraulichkeit und Autorisierung, die in Kapitel 2.2 näher beleuchtet werden. Aufbauend auf diesen grundlegenden lassen sich *komplexere Schutzziele* wie Schutz der Privatsphäre oder Nachvollziehbarkeit definieren.

Um die grundlegenden Schutzziele zu gewährleisten, existieren *kryptografische* Methoden und Routinen um z.B. Daten zu verschlüsseln oder ihre Inte-

grität sicher zu stellen. Diese Methoden werden ebenfalls in Kapitel 2.2 beschrieben. Existierende kryptografische Maßnahmen benötigen außerdem eine *Infrastruktur*, also ein System, das beschreibt, wie zwei oder mehr Kommunikationspartner gemeinsame Methoden und passendes Schlüsselmaterial auswählen und so gesicherte Informationen übertragen können. In Kapitel 6.1 wird ein umfassender Überblick über existierende Infrastrukturen gegeben und ihre Nachteile werden diskutiert.

Anhand der vorgenommenen Klassifikation kann die folgende Einordnung erfolgen. Im weiteren Verlauf dieser Arbeit wird eine **Infrastruktur** zur Erfüllung der **grundlegenden Schutzziele** der **Kommunikationssicherheit** und somit der **Informationssicherheit** vorgestellt.

1.2 Aufbau der Arbeit

Der Aufbau der vorliegenden Arbeit ist in Abbildung 1.2 dargestellt. In Kapitel 2 werden zunächst Grundlagen zu Begriffen und Protokollen der Kryptografie, Service-orientierten Architekturen, DPWS und der Web Service Security Spezifikationssuite diskutiert, da sie in der gesamten Arbeit von Belang sind. Im Zusammenhang mit diesen Grundlagen werden Grundkomponenten für Sicherheitsinfrastrukturen aus klassischen Rechnernetzen beleuchtet und ihre Tauglichkeit für verteilte eingebettete Systeme untersucht. Ein Ergebnis wird sein, dass ein umfassendes Sicherheitskonzept für die anvisierten Szenarien aufgrund der Protokollheterogenität auf Anwendungsebene implementiert sein muss.

Aufbauend auf dieser Erkenntnis geht Kapitel 3 der Frage nach, ob die Verschlüsselung und Integritätssicherung von Daten auf Anwendungsebene gravierende Performance-Unterschiede gegenüber der derzeit am weitesten verbreiteten Alternative – TLS auf Transportebene – aufweist. In Anlehnung an die Basisspezifikation WS Security beschreibt Kapitel 3.2 eine kompakte Alternative namens WS Compact Security. Ein Experiment zum Performance-Vergleich zeigt, dass die im Rahmen dieser Arbeit entstandene Implementierung zwar langsamer ist als TLS, der Faktor zwei bis drei jedoch akzeptabel ist, da dafür keine Abhängigkeit von TLS mehr besteht. Dieser Machbarkeitsnachweis bildet

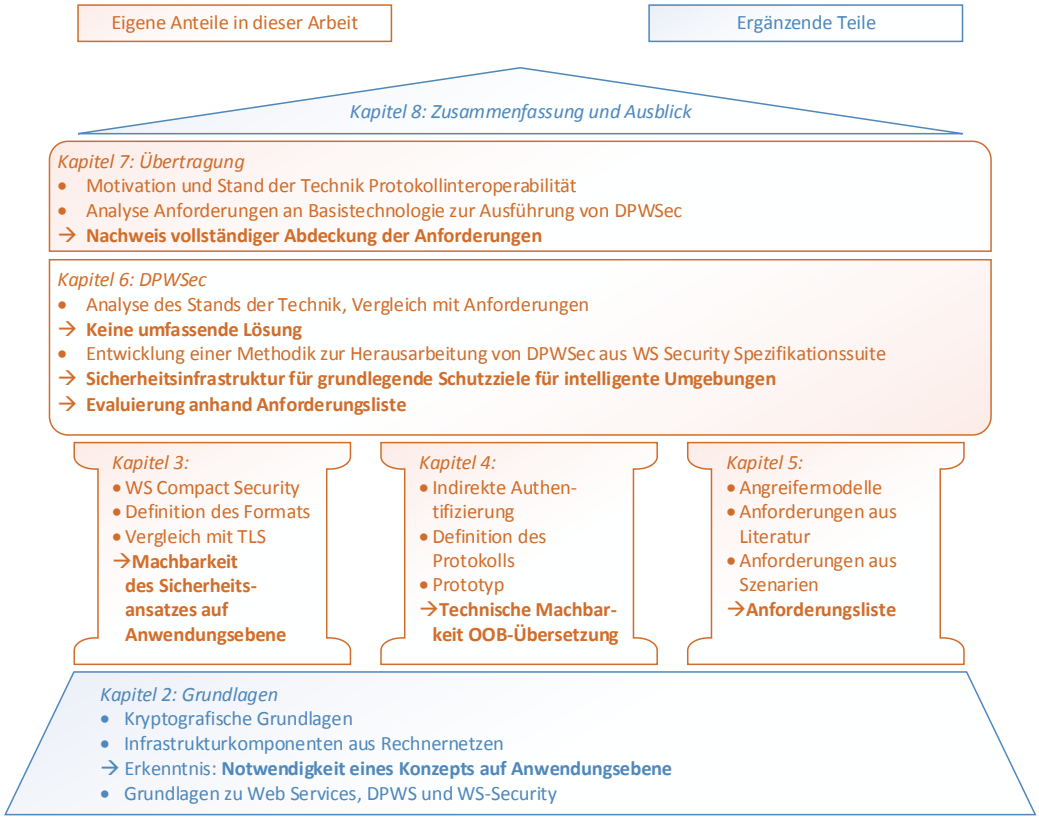


Abbildung 1.2: Aufbau der vorliegenden Arbeit

eine von drei Säulen, auf denen später eine umfassende Sicherheitsinfrastruktur für verteilte eingebettete Systeme fußen wird.

Die zweite Säule stellt die Lösung des Problems dar, wie zwei Geräte im Feld eine Vertrauensbeziehung durch direkten Austausch einer Einweg-Pin aufbauen können, wobei sie aufgrund inkompatibler Bedienelemente keinen gemeinsamen Out-of-Band-Kanal aufbauen können. Kapitel 4 stellt hierfür ein Konzept vor, indem Multimediageräte wie Smartphones, die heute in allen intelligenten Umgebungen vorhanden sind, eine transparente Übersetzung des Out-of-Band-Kanals vornehmen.

Die verbleibende dritte Säule bildet die umfassende Anforderungsanalyse in Kapitel 5. Auf Basis akzeptierter Angreifermodelle, Anforderungen aus der Literatur und konsolidierten Szenarien entsteht in Kapitel 5.4 eine Liste an Anforderungen, die eine Sicherheitsinfrastruktur für verteilte eingebettete Systeme erfüllen muss.

Fußend auf diesen drei Säulen beschreibt Kapitel 6 das DPWSec, eine umfassende Sicherheitsinfrastruktur für verteilte eingebettete Systeme auf Basis von DPWS, nachdem die Analyse des Stands der Technik zeigen konnte, dass bisher keine Lösung allen Punkten der Anforderungsanalyse gerecht werden konnte. Die entwickelte Sicherheitsinfrastruktur basiert auf den Spezifikationen der Web Service Security Suite, die für Desktop-PCs und Serversysteme entworfen wurde. Nach Beschreibung der entwickelten Methodik zur Herausarbeitung der wesentlichen Konzepte wird die Infrastruktur selbst beschrieben. Das Kapitel schließt mit einer Bewertung anhand der Anforderungsliste.

Diese Bewertung kann an dieser Stelle keine Aussagen zu einer möglichen Protokollinteroperabilität treffen. Aus diesem Grund wird im gesonderten Kapitel 7 untersucht, ob und unter welchen Umständen das entwickelte DPWSec auf weitere Basistechnologien neben DPWS übertragen werden kann, um eine sichere Protokollinteroperabilität zu erreichen.

Die Arbeit schließt in Kapitel 8 mit einer Zusammenfassung und Diskussion der Ergebnisse und gibt einen Ausblick auf mögliche weitere Arbeiten.

1.3 Begriffe und Konventionen

An dieser Stelle sollen durchgängig verwendete Konventionen dieser Arbeit geklärt werden.

So wird eine klare Trennung zwischen den Begriffen *Gerät* und *Device* vorgenommen. Ersteres bezeichnet ein tatsächliches, physisches Gerät, wie einen PC oder einen Lichtschalter. Der Terminus Device hingegen bezieht sich auf sein Äquivalent aus der Begriffswelt von DPWS. Ein Device ist demnach eine logische Einheit, die dem Standard DPWS entspricht und Dienste und Operationen anbietet.

Die doppeldeutig genutzte Bedeutung von WS Security kann zu Verwechselungen führen. In der Fachwelt kann dies eine einzelne Spezifikation benennen (vgl. Kapitel 2.4.1) ebenso wie eine ganze Spezifikationssammlung (vgl. Kapitel 2.4), wobei ersteres Teil von letzterem ist. In dieser Arbeit wird *WS Security* als Begriff für die einzelne Spezifikation genutzt, die gesamte Sammlung wird als *WS Security Spezifikationssuite* bezeichnet.

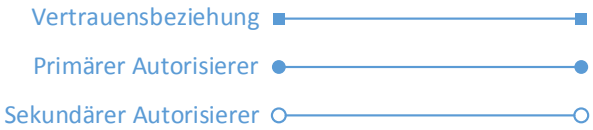


Abbildung 1.3: Legende für Verbindungen zwischen Geräten

Schließlich gilt die folgende Legende für Abbildungen. Wie in Abbildung 1.3 gezeigt, bezeichnet eine Verbindung zwischen zwei Geräten mit quadratischen Endpunkten eine bestehende Vertrauensbeziehung. Runde Endpunkte zeigen an, dass ein Gerät Autorisierungsentscheidungen für das andere treffen darf. Das Konzept des primären Autorisierers (vgl. Kapitel 6.3.6) wird durch gefüllte Kreise dargestellt, die übrigen (sekundären) durch leere Kreise.

Kapitel 2

Grundlagen

Eine umfangreiche Diskussion verbreiteter Sicherheitsinfrastrukturkomponenten und ihrer Tauglichkeit für verteilte eingebettete System sowie die Ableitung, dass eine Sicherheitsinfrastruktur auf Anwendungsebene implementiert werden muss, wurde im Rahmen der Anfertigung dieser Arbeit bereits wie folgt veröffentlicht:

> *Sebastian Unger, Stefan Pfeiffer, Dirk Timmermann: „How much Security for Switching a Light Bulb - The SOA Way“ in 8th International Conference on Wireless Communications and Mobile Computing 2012. ISBN: 978-1-4577-1379-8 [UPT12b]*

In diesem Kapitel werden benötigte Grundlagen und die Teile des Stands der Technik erläutert, die in der gesamten Arbeit von Belang sind. Notwendige Ergänzungen werden im Zusammenhang mit den übrigen Kapiteln der Arbeit vorgenommen.

2.1 Ambiente eingebettete Systeme im Internet der Dinge

Seit Mark Weiser in [Wei91] den Begriff des *Pervasive Computing* nachhaltig prägte, haben unterschiedliche Forschungsrichtungen und Geschäftsmodelle seine Vision aufgegriffen, weiterentwickelt oder abgeändert. Während das Pervasive Computing die Allgegenwärtigkeit von Computern vorhersagte, hat die Strömung der *Ambient Intelligence* vor allem das Ziel, Computer in den

Hintergrund treten zu lassen und uns beinahe unbemerkt durch Assistenzfunktionen das Leben zu erleichtern. Konkrete Ausprägungen sind dieser Tage beispielsweise das intelligente Heim (*Smart Home*), *Smart Metering* oder das *Ambient Assisted Living (AAL)*. Erste Komponenten für Smart Homes wie z.B. intelligente Beleuchtungskonzepte oder Türschlösser sind bereits auf dem Endkundenmarkt erhältlich und bieten die anvisierten Assistenzfunktionen. Ergänzt wird ein Smart Home durch Smart Meter, die den Stromverbrauch seiner Nutzer optimieren und den Versorgern ein detailliertes Planungsinstrument zu Verfügung stellen sollen. AAL kombiniert all dies, legt seinen Fokus jedoch auf ein möglichst selbstbestimmtes Leben auch im hohen Alter. Dies beinhaltet eine unauffällige Überwachung der Nutzer, um Krankheit oder Unfälle rechtzeitig zu erkennen, sowie eine Verbindung zum behandelnden Arzt.

Der gemeinsame Nenner dieser Anwendungsfelder liegt in der Tatsache, dass einfache, alltägliche Gegenstände mit einem gewissen Maß an Intelligenz ausgestattet werden. Das bedeutet, dass sie eine Recheneinheit beinhalten. Um diese Anwendungsfelder umfassend zu erschließen, muss jedoch weiterhin gewährleistet sein, dass die einzelnen Komponenten miteinander kommunizieren können. Erst so lassen sich komplexe Funktionalitäten durch die geschickte Orchestrierung mehrerer einfacher Dienste erreichen. Um diese Kommunikation zu gewährleisten, bedarf es einer Infrastruktur, die verschiedene Eigenschaften erfüllt.

So muss diese Infrastruktur eine globale Kommunikation ermöglichen, um z.B. Wettervorhersagen einzuholen oder Trainings- oder Vitaldaten an die Praxis des behandelnden Arztes zu senden. Eine solche Infrastruktur wurde bereits in den vergangenen Jahrzehnten mit dem Internet aufgebaut, die nun für diesen Zweck genutzt werden soll. Da nun nicht mehr Nutzer mit PCs diesen Kommunikationskanal nutzen, sondern alltägliche „Dinge“, formte sich der Begriff des Internets der Dinge (*Internet of Things, IoT*). Die Verwendung des Internets als zugrundeliegende Infrastruktur bietet einen weiteren inhärenten Vorteil. Das Internet basiert vollständig auf offenen, lizenzfreien und frei zugänglichen Standards, was eine herstellerübergreifende Kommunikation begünstigt.

Den verschiedenen Visionen, Ausprägungen und Geschäftsfeldern ist die eine Sache gemein, dass sie eine hohe Zahl von Geräten beinhalten, die miteinander kommunizieren sollen. Neben den wenigen „klassischen“ Computern befinden

sich darunter auch zahllose Recheneinheiten in alltäglichen Gegenständen, die sogenannten *eingebetteten Systeme*. Sie besitzen häufig nur geringe Ressourcen, da dies einen geringen Stromverbrauch mit sich bringt. Das ist zunächst aus ökologischen Gesichtspunkten wünschenswert, jedoch auch aufgrund von möglicher Hitzeentwicklung, Formfaktor oder Akkulaufzeit. Da sich diese Arbeit darauf konzentriert, eine sichere Infrastruktur für verteilte eingebettete Systeme vorzustellen, werden die hier besprochenen Begriffe wie Intelligente Umgebungen, Ambient Assisted Living oder das Internet der Dinge zum Teil synonym verwendet, auch wenn sich ihre Ausprägungen im Detail unterscheiden.

2.2 Kryptografische Grundlagen

Im folgenden Abschnitt werden zunächst die grundlegenden Schutzziele erläutert. Anschließend werden kryptografische Algorithmen, Routinen und Protokolle beschrieben, die in dieser Arbeit von Bedeutung sind und zur Erfüllung der Schutzziele dienen.

2.2.1 Die grundlegenden Schutzziele

Die vier grundlegenden Schutzziele lauten *Vertraulichkeit*, *Integrität*, *Authentizität* und *Autorisierung*, die Stammberger et al. in [SSAC10] als „die wichtigsten Sicherheitsbelange“ bezeichnen. Sie bilden die Grundlage für höherwertige Schutzziele wie Anonymität und Privacy. Die vier grundlegenden Schutzziele werden im Folgenden zunächst einzeln und anschließend ihr gemeinsames Wirken beschrieben.

Eine Konversation – also ein Austausch von Nachrichten oder Informationen – gilt als vertraulich, wenn sichergestellt ist, dass kein unberechtigter Dritter in der Lage ist, durch Belauschen des Austauschs von Nachrichten Rückschlüsse auf ihren Inhalt zu ziehen. Die Eigenschaft der Integrität ist erfüllt, wenn garantiert werden kann, dass eine Nachricht nicht durch Unbefugte verändert wurde. Das bedeutet auch, dass es nicht möglich sein darf, dass Daten unbemerkt unvollständig empfangen wurden. Diese Eigenschaft ist immer dann von Bedeutung, wenn der Empfänger einer Nachricht eine Aktion basierend auf dem Inhalt dieser Nachricht auslöst. Ebenso wichtig wie die Integrität einer

Nachricht ist ihre Authentizität, also die Tatsache, dass sie einzig und allein von dem genannten Absender stammen kann. Als Autorisierung bezeichnet man das Problem, zu entscheiden, ob der Absender einer Anfrage berechtigt ist, auf die angefragte Ressource zuzugreifen. Häufig muss dabei unterschieden werden, ob lesend oder schreibend auf diese Ressource zugegriffen werden darf.

Integrität und Authentizität sind meist untrennbar miteinander verflochten. Die Authentizität einer Nachricht kann nur bestätigt werden, wenn ihre Integrität gewährleistet ist. Gleichzeitig kann die Integrität einer Nachricht nur überprüft werden, wenn die Authentizität des Absenders zweifelsfrei bestimmt werden kann. Erst durch Sicherstellung von Authentizität und Integrität kann ein Autorisierungssystem Entscheidungen auf Basis der Identität des Anfragestellers treffen. Auch, wenn ein wirksamer Zugriffsschutz die Basis für Vertraulichkeit darstellt ([Pö10]), so kann diese nur dann vollständig gewährleistet werden, wenn zusätzliche Maßnahmen das Belauschen berechtigter Anfragen verhindern.

2.2.2 Kryptografische Algorithmen und Protokolle

In diesem Abschnitt werden kryptografische Algorithmen und Protokolle erläutert, die der Erfüllung der vier Schutzziele dienen.

Verschlüsselung zur Wahrung der Vertraulichkeit

Zur Wahrung der Vertraulichkeit von Informationen können diese verschlüsselt werden. Im Folgenden werden gewünschte Eigenschaften und Arten von Verschlüsselungsalgorithmen diskutiert.

Eigenschaften Verschlüsselung lässt sich als Abbildung betrachten. Mit ihr kann eine Klartextnachricht m aus der Menge aller Klartextnachrichten M durch eine Funktion f auf einen Chiffretext c aus der Menge aller Chiffretexte C abgebildet werden. Die abbildende Funktion muss dabei eineindeutig sein, damit zu jeder Nachricht $m \in M$ genau eine Chiffre $c \in C$ entstehen kann und die Umkehrfunktion f^{-1} auf genau eine Chiffre c die ursprüngliche Nachricht m abbildet. Es gilt nach [SSP08]

$$f : M \to C, \qquad m \in M, c \in C \tag{2.1}$$

$$f^{-1} : C \to M, \qquad c \in C, m \in M \tag{2.2}$$

Symmetrische und asymmetrische Chiffren Symmetrische kryptografische Verfahren zeichnen sich dadurch aus, dass Ver- und Entschlüsselung einer Nachricht m bzw. einer Chiffre c mit dem identischen Schlüssel k durchgeführt werden. Dieser muss von den Parteien streng geheim gehalten werden. Daher bezeichnet man diese Schlüssel häufig als *Shared Secret*. Die Funktionen zur Ver- und Entschlüsselung erhalten als Parameter nicht nur den jeweiligen Klar- bzw. Chiffretext sondern darüber hinaus auch den Schlüssel. Neben der vollständigen Schreibweise $c = f(k, m)$ bzw. $m = f^{-1}(k, c)$ sind auch die kompakten Schreibweisen $c = f_k(m)$ bzw. $m = f_k^{-1}(c)$ gängig. Es gilt nach [SSP08]:

$$c = f(k, m) = f_k(m) = f_k(f_k^{-1}(c)) \tag{2.3}$$

$$m = f^{-1}(k, c) = f_k^{-1}(c) = f_k^{-1}(f_k(m)) \tag{2.4}$$

Ein praktischer Nachteil symmetrischer Verschlüsselung ist das Problem des Austauschs des Schlüssels, falls noch kein sicherer Kanal existiert, über den dieser Schlüssel ohne Zugriff für Dritte übertragen werden könnte. Abhilfe schaffen hier asymmetrische Verschlüsselungsverfahren, bei denen jeder Kommunikationspartner einen öffentlichen und einen geheimzuhaltenden Schlüssel besitzt. Sie bedienen sich der Eigenschaft bestimmter mathematischer Funktionen, dass diese Funktionen selbst zwar leicht auszuführen sind, ihre Umkehroperationen jedoch mit derzeit verfügbaren Mitteln praktisch undurchführbar sind. Beispiele für solche mathematischen Funktionen sind die Multiplikation großer Primzahlen (mit Primfaktorzerlegung als Umkehroperation) oder die Umkehrung des diskreten Logarithmus ([SSP08]). Diese Eigenschaft gewährleistet unter bestimmten Bedingungen, dass ein Teilnehmer einen geheimen Schlüssel d frei wählen kann, aus dem er mit der Funktion g den öffentlichen Schlüssel e einfach berechnen kann, sodass für d und e gilt:

$$e = g(d) \tag{2.5}$$

$$c = f_e(m) = f_e(f_d(c)) \tag{2.6}$$

$$m = f_d(c) = f_d(f_e(m)) \tag{2.7}$$

Da die Umkehrfunktion von g, also g^{-1} praktisch nicht berechenbar ist, kann ein öffentlicher Schlüssel frei zugänglich gemacht werden. Ein Angreifer kann daraus keine Rückschlüsse auf den privaten Schlüssel ziehen, da $d = g^{-1}(e)$ nicht berechenbar ist. Wollen zwei Parteien A und B nun verschlüsselt kommunizieren, nutzen sie dazu den öffentlichen Schlüssel des jeweiligen Gegenübers, der über einen ungesicherten Kanal ausgetauscht werden darf.

Strom- und Blockchiffren Perfekte Sicherheit kann eine Verschlüsselung nur dann bieten, wenn ein Schlüssel genutzt wird, der die gleiche Länge besitzt wie die zu verschlüsselnde Nachricht und wenn dieser Schlüssel gleichverteilt zufällig erzeugt bzw. ausgewählt wurde. In diesem Fall reicht eine Exklusiv-Oder-Verknüpfung von Klartext und Schlüssel aus, da, wenn beide unbekannt sind, jede mögliche Kombination aus Schlüssel und Klartext zu einem Chiffretext geführt haben kann. Da der Austausch großer Mengen gleichverteilten Schlüsselmaterials nicht praktikabel ist, werden zwei alternative Möglichkeiten genutzt. Zunächst wird ein vergleichsweise kurzer Schlüssel gewählt. Daraus wird dann entweder mit einem bestimmten Verfahren ein quasizufälliger endloser Schlüsselstrom erzeugt (Stromchiffren) oder man nutzt ein aufwändigeres Verfahren als eine Exklusiv-Oder-Verknüpfung um Schlüssel und Klartext miteinander zu verrechnen (Blockchiffren) ([BNS10]). Bei letzterem wird der Klartext zunächst in Blöcke unterteilt. Diese werden dann nicht nur mit dem geheimen Schlüssel, sondern unter Umständen auch mit dem vorhergehenden Block verrechnet, was statistische Angriffe stark erschwert.

Authentizität, Integrität und Verbindlichkeit

In Abschnitt 2.2.1 wurde gezeigt, dass Nachrichtenauthentizität und -integrität untrennbar miteinander verknüpft sind. Entsprechend werden beide Schutzziele zumeist durch die selben Mechanismen in einem Zuge gewährleistet ([BNS10]).

Message Authentication Codes Ein *Message Authentication Code (MAC)* ist ein „kryptografischer Fingerabdruck“ einer Nachricht, also eine kurze Prüfinformation zu einem langen Klartext, die nur vom Besitzer eines Schlüssels berechnet oder überprüft werden kann ([SSP08]). Ein solcher MAC stellt demnach sicher, dass

a) der MAC mit Sicherheit vom Besitzer des Geheimnisses berechnet wurde (Authentizität) und

b) die Nachricht nicht verändert wurde (Integrität).

Ein Angreifer kann eine abgefangene Nachricht nicht verändern, da der MAC sonst nicht mehr passt. Neuberechnen kann der Angreifer den MAC ebenfalls nicht, da er nicht im Besitz des Geheimnisses ist. Authentizität und Integrität sind demnach gewährleistet.

Neben der MAC-Berechnung auf Basis symmetrischer Blockchiffren spielt insbesondere der Algorithmus keyed-Hash Message Authentication Code (HMAC) ([KBC97, Nat08]) eine wichtige Rolle. Hierbei werden der geheime Schlüssel, die zu sichernde Nachricht und zwei Konstanten auf bestimmte Weise mit einer beliebigen Hash-Funktion verrechnet. Durch die Austauschbarkeit ist auch die Länge des entstehenden MAC im gewissen Rahmen flexibel.

Signaturen Bei den MACs handelt es sich um ein symmetrisches Verfahren. Soll asymmetrische Kryptografie genutzt werden, um den Austausch von Schlüsseln zu vereinfachen, werden wie zuvor beschrieben kryptografische Signaturen eingesetzt. Der Versender einer Nachricht m bildet den Hashwert dieser Nachricht $h(m)$, verschlüsselt diesen mit seinem geheimen Schlüssel d zu einer kryptografische Signatur und hängt diese an die Nachricht an. Der Empfänger der Nachricht m' entschlüsselt die Signatur mit dem öffentlichen Schlüssel des Absenders und bildet den Hashwert der empfangenen Nachricht $h(m')$. Stimmen $h(m)$ und $h(m')$ überein, sind Authentizität und Integrität der Nachricht gewährleistet (vgl. Abb. 2.1) ([Beu09]). Signaturen besitzen eine Eigenschaft, die MACs nicht innehaben, die sogenannte Verbindlichkeit oder Nichtabstreitbarkeit. In einer Gruppe von Kommunikationsteilnehmern kann ein MAC von jedem erzeugt werden, der im Besitz eines gemeinsamen Geheimnisses ist. Eine Signatur hingegen kann zwar von jedem Teilnehmer überprüft

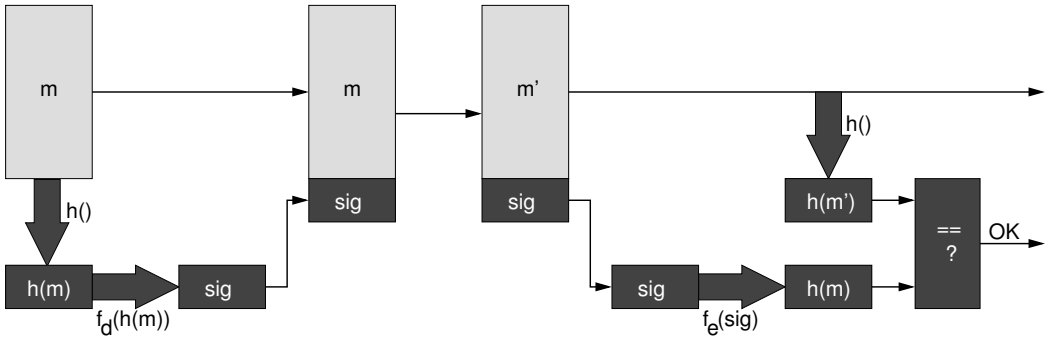


Abbildung 2.1: Grundlegende Funktionsweise kryptografischer Signaturen

werden, ihre Erstellung ist jedoch ausschließlich dem Besitzer des zugehörigen privaten Schlüssels vorbehalten. Während ein Teilnehmer demnach in der Lage ist, eine MAC im Namen seines Gegenübers zu erstellen, gelingt ihm das mit einer Signatur nicht. Sie muss vom angegebenen Absender stammen.

Message Freshness und Replay-Angriffe

Zwei Eigenschaften, die unmittelbar zusammenhängen, sind die Anfälligkeit für *Replay-Angriffe* und die *Message Freshness*. Ein Angreifer hat die Möglichkeit, gesicherten Nachrichtenverkehr zu belauschen und einzelne Nachrichten abzuspeichern. Auch, wenn er keinen vollständigen Zugriff auf den verschlüsselten Inhalt der Nachrichten hat, kann es passieren, dass der Angreifer durch Beobachtung der Reaktionen auf den Erhalt einer Nachricht Rückschlüsse auf dessen Inhalt zieht. Möchte der Angreifer zu einem späteren Zeitpunkt die selbe Aktion provozieren (z.B. Öffnen des Garagentors), könnte er eine Kopie der gespeicherten Nachricht einspielen, ohne ihren Inhalt zu kennen (Replay). Der Empfänger kann nicht erkennen, dass diese Nachricht nicht vom angegebenen Absender stammt. Verhindern lassen sich Replay-Angriffe durch die Gewährleistung der „Frische“ (Freshness) von Nachrichten. Beutelspacher et al. nennen in [BNS10] zwei Mechanismen zur Vermeidung solcher Angriffe. Eine Möglichkeit besteht in der Einbettung von Zeitstempeln in die Nachricht und eine Einigung über ihre Lebensdauer. Das setzt jedoch synchronisierte Uhren zwischen Teilnehmern voraus und bietet nur einen schwachen Schutz, da eine Zeitspanne kein Garant für eine einmalige Nutzung ist. Als Alternative schlagen die Autoren Einmalzufallszahlen *Nonces* vor, die in der Antwort des Empfängers aufgegriffen werden, was im Wesentlichen einem Challenge-

Response-Protokoll entspricht. Eine weitere Möglichkeit besteht in der Verwendung von Sequenznummern ([BMC+05]). Sie sind ein Spezialfall von Nonces und müssen streng monoton steigend sein. So muss ein Empfänger nur prüfen, ob die aktuelle Sequenznummer eines Absenders größer ist als die letzte. Die letzten beiden Verfahren bedürfen einer Speicherstelle für die zuletzt genutzte Zufallszahl oder ID, die den Neustart eines Geräts überdauert.

Key Derivation Functions (KDF)

Haben zwei Parteien ein gemeinsames Geheimnis aufgebaut, könnte es direkt zur Verschlüsselung und zur Berechnung von MACs eingesetzt werden. Es ist jedoch ratsam, stattdessen weitere Schlüssel abzuleiten ([OAS07a]). Insbesondere sollten sich die Schlüssel für die Verschlüsselung und für die MAC-Bildung unterscheiden ([BBB+11]). Dies erschwert laut Snader „bestimmte Angriffe“ [Sna05], zu denen vor allem statistische Angriffe zählen. Aus diesem Grund kommen *Key Derivation Functions (KDFs)* zum Einsatz. Sie errechnen mit einer festgelegten Pseudozufallszahlenfunktion (*Pseudo Random Function, PRF*) aus einem Seed und dem eigentlichen Shared Secret einen abgeleiteten Schlüssel ([WI13]). Das NIST definiert hierzu in [Che09] Verfahren für KDFs, abstrahiert von einer möglichen PRF. Ergänzend dazu werden in [BK12] PRFs empfohlen. Eines davon sind die zuvor beschriebenen HMAC-Verfahren, die auch für TLS in [DA99] und für Web Services in [OAS07a] festgelegt werden.

Authentifizierungsverfahren und Schlüsselaustausch

Um die in Abschnitt 2.2.1 beschriebene Eigenschaft der Authentizität einer Nachricht überprüfen zu können, muss zuvor eine Authentifizierung infragekommender Absender erfolgen. Authentifizierung ist ein Prozess, der sich aus den Bestandteilen Identifikation und Schlüsselaustausch zusammensetzt. Die Identifikation stellt über eine geeignete Maßnahme sicher, dass die tatsächliche Identität des Gegenübers derjenigen entspricht, die er angibt. Die Authentizität des Gegenübers kann zumeist nur für die Dauer des Authentifizierungsvorgangs sichergestellt werden. Daher wird während der Authentifizierung Schlüsselmaterial ausgetauscht, aufgebaut oder verifiziert (nachfolgend als „austauschen“ zusammengefasst). Erst durch diesen Schritt und der damit einhergehenden Verknüpfung einer Identität mit kryptografischem Schlüsselmaterial, wie einem

Shared Secret oder einem Zertifikat, kann das Ergebnis einer Authentifizierung dauerhaft gehalten werden.

Das Ergebnis eines erfolgreichen Authentifizierungsvorgangs ist eine Vertrauensbeziehung zwischen zwei Geräten. In Abgrenzung zu Arbeiten insbesondere aus dem Kontext drahtloser Sensornetze, in denen Vertrauen in ein Gerät einen beliebigen reellen Wert zwischen 0 und 1 annehmen kann (vgl. z.B. [CFZA02, CAMN+02]), wird in dieser Arbeit der Begriff Vertrauensbeziehung als binärer Zustand im Sinne einer positiven, dauerhaften Identifikation eines Gegenübers verstanden.

Man unterscheidet grundsätzlich zwei Arten von Authentifizierungsvorgängen, direkte und delegierte. Während der direkten Authentifizierung führen zwei Parteien ohne jeden Mittelsmann ein Protokoll durch, das dazu führt, dass sich genau diese beiden Parteien gegenseitig identifizieren und dabei Schlüsselmaterial austauschen. Nimmt man Vertrauensbeziehungen als transitiv an, gilt also

$$\tau(u, v) \wedge \tau(v, w) \Rightarrow \tau(u, w), \tag{2.8}$$

können auf Basis direkter Vertrauensbeziehungen delegierte Authentifizierungen durchgeführt werden. Ist Teilnehmer v in Formel 2.8 eine sogenannte *vertrauenswürdige dritte Instanz*, so kann sie zuvor direkt aufgebaute Vertrauensbeziehungen vermitteln. Beispiele für direkte Authentifizierungsverfahren sind zum Beispiel das werksseitige Installieren von Zertifikaten oder Challenge-Response-Protokolle. Vertreter delegierter Authentifizierung sind Public-Key-Infrastrukturen, das Kerberos-Protokoll oder WS Trust. Die genannten Verfahren werden im Verlauf dieses Kapitels beschrieben.

Zertifikate & Public-Key-Infrastrukturen Ein Zertifikat bezeichnet im Allgemeinen „ein digitales Dokument, in dem eine Instanz [...] einen bestimmten Sachverhalt mittels digitaler Signatur bestätigt“ [BMC+05]. Praktisch bezeichnet es ein Dokument, das neben Angaben zu einer Entität (Name, Herkunft, etc...) dessen öffentlichen Schlüssel enthält. Die signierende Instanz kann die Entität selbst (*self-signed*) oder eine dritte Partei sein. Auf Basis von Zertifikaten lassen sich sowohl direkte als auch delegierte Authentifizierungen durchführen. Ersteres lässt sich erreichen, indem das self-signed Zertifikat eines Gegenübers vor Ausbringung ins Feld fest in den Speicher eines Geräts

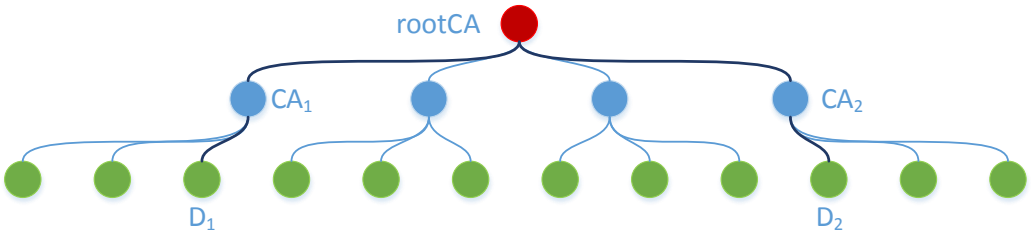


Abbildung 2.2: Schematische Darstellung einer Zertifikatshierarchie

installiert wird. Dieser Umstand macht das Zertifikat vertrauenswürdig. Da es in den wenigsten Fällen möglich ist, alle zukünftigen Kommunikationspartner vor der Ausbringung ins Feld zu kennen, wird stattdessen oft eine *Public Key Infrastruktur* (PKI) für delegierte Authentifizierungen aufgebaut. Eine hierarchische PKI auf Basis von Zertifikaten ist in Abbildung 2.2 dargestellt. Bei Verbindungsaufbau bezieht das Gerät D_1 das Zertifikat C_2 von Gerät D_2. Dieses ist von der *Certificate Authority (CA)* CA_2 signiert. Kennt und vertraut D_1 CA_2 (durch Vorinstallation), so kann es die Signatur im Zertifikat überprüfen und die delegierte Authentifizierung ist erfolgreich beendet. Andernfalls muss D_1 die Hierarchie der Infrastruktur weiter in Richtung der Wurzel iterieren, bis es eine CA findet, der es vertraut. Die Suche muss spätestens beim Erreichen der Wurzel, der *root CA*, erfolgreich sein. Sie besitzt ein self-signed Zertifikat, das mindestens vorinstalliert sein muss, damit das beschriebene Protokoll funktioniert.

Das Kerberos-Protokoll Das Kerberos-Protokoll stellt ein Beispiel für eine zentralisierte Authentifizierungsinfrastruktur ohne asymmetrische Kryptografie dar. Seine Grundzüge wurden 1978 von NEEDHAM und SCHROEDER entwickelt ([NS78]). In den 1980er Jahren wurde das Protokoll am MIT verbessert ([SSP08]), bevor es in seiner heutigen Form entstand ([NYHR05]).

Vorausgesetzt wird, dass alle Teilnehmer $\{A, B, ...\}: X$ in einem Netzwerk bereits eine Vertrauensbeziehung zu einer Trusted Third Party S besitzen und jeder Teilnehmer einen symmetrischen Schlüssel mit S k_{XS} ausgetauscht hat. Will nun Teilnehmer A mit Teilnehmer B kommunizieren (vgl. Abb. 2.3), so sendet er eine Authentifizierungsanfrage mit den Identifikationen von sich selbst und seinem Ziel an S. Dieser antwortet mit zwei Nachrichten. Beide enthalten einen Timestamp, eine Identifikation sowie den symmetrischen Schlüssel

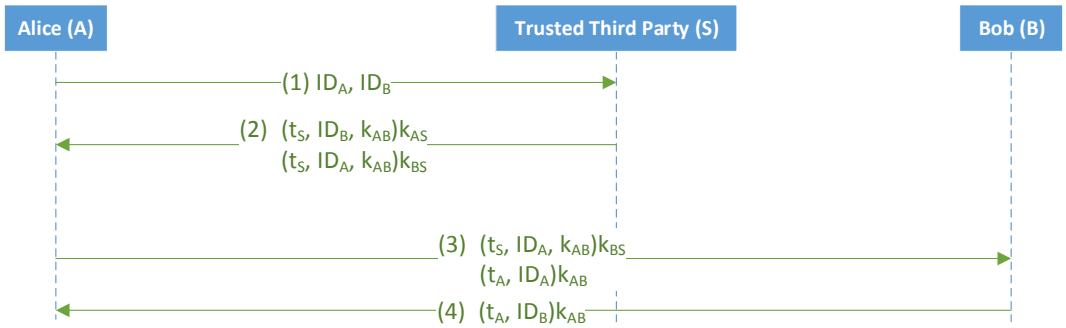


Abbildung 2.3: Schematischer Ablauf des Kerberos-Protokolls nach [SSP08]

für A und B k_{AB}. Jedoch ist die erste Nachricht für A mit k_{AS} und die zweite mit k_{BS} für B verschlüsselt. Um mit B zu kommunizieren, leitet A die zweite Nachricht zunächst an B weiter. Nur B kann den Inhalt entschlüsseln und erlangt so das notwendige Schlüsselmaterial für die Kommunikation A k_{AB}. Abschließend kann A nun verschlüsselte Nachrichten an B senden, da beide über das notwendige Schlüsselmaterial verfügen und die Identität des jeweiligen Gegenübers durch S authentifiziert wurde.

Direkte Authentifizierung Eine Möglichkeit der direkten Authentifizierung besteht im *Out-of-Band (OOB)* Austausch eines Einmalpassworts bzw. einer Einwegpin. Können zwei Parteien einander nachweisen, dass sie im Besitz der festgelegten Pin sind, haben sie sich zweifelsfrei gegenseitig identifiziert. Die Schwierigkeit liegt hier im Nachweis über das Wissen um die Pin, ohne sie einem möglicherweise boshaften Gegenüber zu verraten. Hierzu kommen *Challenge-Response-Protokolle* zum Einsatz. Die Parteien senden sich gegenseitig Zufallszahlen zu (Challenges) die der jeweils andere mit der Einwegpin zu seiner Response verschlüsselt. Einem Angreifer ist es nicht möglich, aus der Kenntnis um Challenge und Response Rückschlüsse auf den verwendeten Schlüssel zu ziehen. Die legitimen Teilnehmer können jedoch durch Kenntnis der Pin verifizieren, dass die Response des Gegenüber zur eigenen Challenge passt. Hierbei handelt es sich jedoch nur um eine Identifizierung, nicht um einen Schlüsselaustausch. In [UZG+10] wird ein Protokoll vorgeschlagen, das im Rahmen eines Challenge-Response-Protokolls gleichzeitig self-signed Zertifikate der Parteien austauscht, um die direkte Authentifizierung zu vervollständigen.

Schlüsselaufbau mit Diffie-Hellman Bei dem Diffie-Hellman-Protokoll ([DH76]) handelt es sich zunächst um eine Möglichkeit, ein gemeinsamen Geheimnis zwischen zwei Parteien aufzubauen, ohne es übertragen zu müssen. Es basiert auf den mathematischen Eigenschaften von Primzahlen und Galois-Körpern und soll an dieser Stelle vereinfacht behandelt werden. Ausführliche mathematische Diskussionen finden sich z.B. in [DH76, Eck08].

Die beiden Parteien Alice und Bob einigen sich öffentlich auf eine Primzahl p und eine Basis g aus dem Galois-Körper $GF(p)$. Alice und Bob wählen je eine Zufallszahl a bzw. b als geheime Schlüssel und berechnen daraus jeweils ihre öffentlichen Schlüssel α bzw. β:

$$\alpha = (g^a)\ mod\ p, \qquad a \in [1, p-2] \tag{2.9}$$

$$\beta = (g^b)\ mod\ p, \qquad b \in [1, p-2] \tag{2.10}$$

Tauschen Alice und Bob anschließend ihre öffentlichen Schlüssel aus, so sind sie in der Lage, das identische gemeinsame Geheimnis k zu berechnen:

$$k = (\beta^a)\ mod\ p = (g^b)^a\ mod\ p = g^{b \cdot a}\ mod\ p \tag{2.11}$$

$$k = (\alpha^b)\ mod\ p = (g^a)^b\ mod\ p = g^{a \cdot b}\ mod\ p \tag{2.12}$$

Ein Angreifer, der diesen Austausch belauscht, hat keine Möglichkeit, k zu berechnen, da er keine Rückschlüsse auf die privaten Schlüssel ziehen kann ([SSP08]). Allerdings kann er sich während des Verbindungsaufbaus als Man-In-The-Middle zwischen Alice und Bob schalten und vortäuschen, der jeweils andere zu sein. So baut er unbemerkt einen Schlüssel k_A mit Alice und einen Schlüssel k_B mit Bob auf, da die Teilnehmer durch das Diffie-Hellman-Protokoll nicht authentifiziert werden. Es findet demnach nur der Schlüsselaustausch statt, jedoch nicht die gegenseitige Identifikation.

Ein weiterer Mechanismus zur direkten Authentifizierung auf Basis eines authentifizierten Diffie-Hellman-Schlüsselaufbaus sowie Vor- und Nachteile direkter und delegierter Authentifizierung werden in Kapitel 4 besprochen.

2.2.3 Sicherheitsmaßnahmen im Netzwerk-Stack

Konkrete Sicherheitsinfrastrukturkomponenten lassen sich auf jeder Schicht des TCP/IP-Referenzmodells einrichten, das eine kondensierte Fassung des

ISO-OSI-Referenzmodells darstellt (vgl. Abb. 2.5, S. 29). Im Folgenden werden die wichtigsten Infrastrukturkomponenten für Rechnernetze unterhalb der Anwendungsebene beschrieben.

MAC-Layer

Drahtlose Übertragungstechnologien haben das Einrichten von Netzwerken insbesondere im Heimbereich deutlich vereinfacht, da keine zusätzlichen Kabel verlegt werden müssen. Drahtlosen Netzwerken fehlt jedoch eine inhärente Sicherheitseigenschaft der drahtgebundenen Netzwerke, nämlich der eingeschränkte physische Zugriff. Bei drahtgebundenen Netzwerken muss sich ein Angreifer direkten, physischen Zugriff auf das Netzwerk verschaffen, z.B. durch Einbruch. Bei drahtlosen Netzwerken ist diese Eigenschaft nicht gegeben. Angriffe auf Funknetze lassen sich auch außerhalb der Wohnung oder sogar des Hauses begehen.

Aus diesem Grund werden insbesondere Funknetzwerke schon auf Ebene der Sicherungsschicht verschlüsselt, um die Sicherheitseigenschaft drahtgebundener Netzwerke nachzubilden. Das erste im Heimbereich eingesetzte Verschlüsselungsprotokoll zur Sicherung von WLAN-Netzwerken heißt deswegen auch *Wired Equivalent Privacy (WEP)*.

Der hauptsächliche Einsatzzweck von WLAN ist, PCs, Notebooks und Smartphones miteinander zu vernetzen. Das Hauptaugenmerk liegt hierbei auf dem Datendurchsatz. Das Funkprotokoll IEEE 802.15.4 hingegen wurde geschaffen, um kleine, rechenschwache und üblicherweise batteriebetriebene Geräte drahtlos zu vernetzen. Solche Geräte finden sich in intelligenten Umgebungen vor allem in Form kleiner Sensoren und Aktoren, die mobil und daher batteriebetrieben und über Funktechnologien erreichbar sein sollen. Hierbei liegt die Priorität der Funkschnittstelle also nicht beim Datendurchsatz sondern bei einer möglichst niedrigen Leistungsaufnahme.

WLAN: WEP - WPA2 Enterprise Wie eingangs erwähnt, galt es im Zuge des Aufkeimens des Einsatzes drahtloser Netzwerke, die inhärente Sicherheitseigenschaft drahtgebundener Netzwerke nachzubilden. Dazu gehört auch, dass jeder Teilnehmer eines Netzwerks alle Pakete innerhalb des Netzwerks

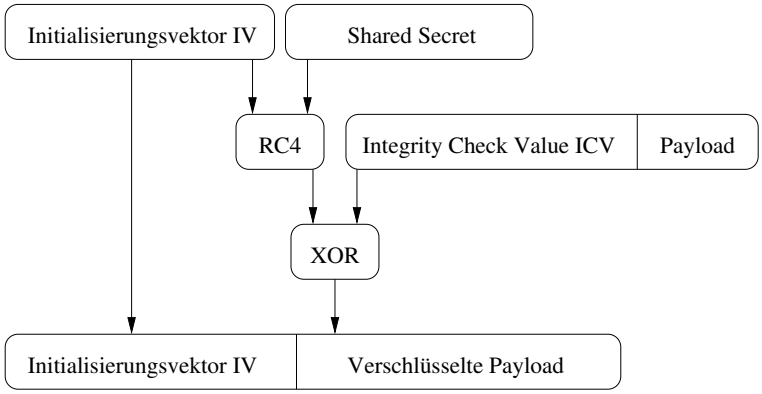


Abbildung 2.4: Funktionsweise von WEP nach [SSP08]

abhören kann. So auch bei dem Verschlüsselungsprotokoll Wired Equivalent Privacy, dessen Funktionsweise in Abbildung 2.4 dargestellt ist.

Der Stromchiffren-Algorithmus RC4 wird hierbei als Pseudo-Zufallszahlengenerator verwendet. Dieser generiert quasizufällige Bitsequenzen, mit denen das eigentliche Datenpaket und eine Prüfsumme, der Integrity Check Value (ICV), per XOR verknüpft werden. Zur Initialisierung des Zufallszahlengenerators wird hierbei ein öffentlicher, im Klartext übertragener, 24 Bit langer Initialisierungsvektor (IV) sowie das Shared Secret genutzt. Das fertige Datenpaket enthält den IV und die verschlüsselte Payload. Fluhrer et al. zeigten bereits 2001 in [FMS01], wie WEP wegen seines kurzen Initialisierungsvektors und des sich nicht ändernden geheimen Schlüssels durch spezielle Replay-Angriffe gebrochen werden kann. WEP besitzt deswegen heute keine praktische Relevanz mehr.

Nachdem WEP keine wirksame Verschlüsselung mehr bieten konnte, begann die WiFi-Alliance mit ihrer Arbeit an dem Standard IEEE 802.11i, der eine neue Sicherheitsarchitektur für WLANs bieten sollte. Um jedoch schnell auf die Tatsache zu reagieren, dass WEP keinen wirksamen Schutz mehr bieten konnte, wurde gleichzeitig an einer Zwischenlösung gearbeitet. Als besondere Anforderung galt, dass die neue Lösung auf bereits im Endkundenumfeld eingesetzter Hardware – also lediglich durch ein Softwareupdate – einsetzbar sein sollte. Die Lösung wurde mit dem Protokoll Wireless Protected Access (WPA) gefunden. WPA basiert im Wesentlichen auf den Mechanismen von WEP. Ein zusätzlicher Message Integrity Code (MIC) verhindert jedoch die unter [FMS01] beschriebenen Replay-Angriffe. Außerdem wird nicht mehr das

Shared Secret direkt als Initialisierung für die Zufallszahlen genutzt, sondern das *Temporal Key Integrity Protocol (TKIP)* errechnet für jedes Datenpaket einen neuen Schlüssel. Zusätzlich mit einem erweiterten Initialisierungsvektor wird damit verhindert, dass mehrmals der gleiche Schlüssel mit dem gleichen Initialisierungsvektor verwendet wird, was – wie bei WEP – Autokorrelationsangriffe ermöglichen würde.

Die Spezifikation IEEE 802.11i, die eine neue Sicherheitsarchitektur für WLANs festlegt, ging schließlich im Standard IEEE 802.11-2007 [IEE07] auf. Das Sicherheitsprotokoll WPA2 implementiert diese Spezifikation und wird daher als Nachfolger von WPA gehandelt. Der wesentliche Unterschied zu WPA liegt in der Nutzung von *Advanced Encryption Standard (AES)* als Verschlüsselungsalgorithmus. Dieser bietet eine stärkere Verschlüsselung als RC4.

Da WPA2 einen anderen Algorithmus verwendet als WPA, besteht keine Kompatibilität mehr. Ein Softwareupdate wie beim Umstieg von WEP auf WPA ist demnach nicht ausreichend, da neue Hardware zur Implementierung von AES notwendig ist. Nachdem jedoch erste praktische Angriffe auf WPA bekannt sind, wird empfohlen, gänzlich auf WPA2 umzusteigen [SSP08, Eck08, TB09, OM09].

Abschließend sei erneut darauf hingewiesen, dass die Protokolle WEP, WPA und WPA2 zur Aufgabe haben, die inhärente Sicherheitseigenschaft eines drahtgebundenen Netzwerks wiederherzustellen. Insbesondere bei WEP bedeutet das, dass alle Teilnehmer eines Netzwerks mit dem gleichen Shared Secret alle Datenpakete aller anderen Teilnehmer mithören können (so, wie es auch in drahtgebundenen Netzwerken der Fall ist). Zwar werden bei WPA und WPA2 unterschiedliche Schlüssel für jedes Datenpaket genutzt, diese lassen sich jedoch leicht berechnen, wenn der initiale Handshake beim Verbindungsaufbau mitgehört werden kann.

IEEE 802.15.4 (6LowPAN und ZigBEE) WLAN und dessen Sicherheitsarchitektur wurden als Möglichkeit zur drahtlosen Kommunikation zwischen Computern wie PCs geschaffen. Hierbei steht ein möglichst hoher Datendurchsatz im Vordergrund. In den Domänen der drahtlosen Sensornetze und der Heimautomation hingegen sollen Geräte zum Einsatz kommen, die einen möglichst kleinen Formfaktor besitzen und einen sehr geringen Leistungsbe-

darf haben, da sie häufig batteriebetrieben sind. Hier steht nicht mehr der Datendurchsatz sondern der Leistungsverbrauch im Vordergrund. Aus diesem Grund wurde die Funkschnittstelle IEEE 802.15.4 ([IEE06]) spezifiziert, die sich durch eben diese Eigenschaften auszeichnet. Aufgrund des Einsatzes auf stark ressourcenbeschränkten Geräten, sind auch die in IEEE 802.15.4 verankerten Sicherheitsmaßnahmen nicht so umfassend wie bei WLAN. Sie werden im Folgenden kurz beschrieben.

Grundsätzlich unterstützt IEEE 802.15.4 vier Modi, um Nachrichten kryptographisch zu sichern. Der einfachste Modus lautet *Null* und besagt, dass keinerlei kryptografische Methoden angewandt werden. Im Modus *AES-CTR* werden alle Nachrichten mit AES verschlüsselt. Die Modi *AES-CBC-MAC-32*, *AES-CBC-MAC-64* und *AES-CBC-MAC-128* fügen der Nachricht einen 32, 64 oder 128 Bit langen *Message Authentication Code (MAC)* bei, der die Herkunft der Nachricht authentifiziert und gleichzeitig die Nachrichtenintegrität sicherstellt. Die Kombination aus Verschlüsselung mittels AES und MACs findet sich in den Modi *AES-CCM-32*, *AES-CCM-64* und *AES-CCM-128* wieder ([SW04]).

Die Spezifikation IEEE 802.15.4 legt nicht fest, wie Schlüssel ausgehandelt oder verwaltet werden. Diese Aufgabe müssen höhere Protokollschichten übernehmen ([XCS+06]), womit auch alle Authentifizierungen zwischen Geräten z.B. durch die Anwendungsschicht erfolgen müssen. Dieses Verhalten lässt eine freie Konfiguration der zu verwendenden Schlüssel zu. Sastry und Wagner beschreiben in [SW04] lediglich die folgenden Varianten als die gängigsten. Bei *Network shared keying* nutzen alle Teilnehmer des Netzwerks den gleichen Schlüssel. Somit ist der Verwaltungsaufwand so gering wie möglich, da jedes Gerät nur einen Schlüssel vorhalten muss. Wird dieser Schlüssel jedoch kompromittiert, ist der Aufwand, die Sicherheit wieder herzustellen sehr hoch. Den Gegensatz bildet das *Pairwise keying*. Jedes Gerätetupel verwendet dabei einen eigenen Schlüssel. Das erhöht zwar den Verwaltungsaufwand, wird jedoch ein Schlüssel kompromittiert, ist nur diese eine Verbindung gefährdet und die Sicherheit kann durch Austausch des Schlüssels wiederhergestellt werden. Einen Kompromiss bietet *Group keying*, um den Aufwand der Schlüsselverwaltung und das Maß an Sicherheit auszubalancieren. Als *Hybrid approaches* werden

beliebige Kombinationen der vorigen drei Ansätze, auch gleichzeitig innerhalb eines Netzwerks, zusammengefasst.

Network shared keying zeigt das gleiche Verhalten wie WEP bzw. unter bestimmten Voraussetzungen auch WPA und WPA2, nämlich, dass jeder legitime Netzteilnehmer in Reichweite alle gesendeten Pakete mithören kann. Das lässt sich durch das Pairwise Keying unterbinden. Soll jedoch ein Routing auf IP-Ebene über mehrere Hops stattfinden, muss jeder Hop die Nachricht entschlüsseln um die Ziel-IP-Adresse lesen zu können.

IPsec

IPsec dient der Verschlüsselung, Authentifizierung und Integritätsprüfung von IP-Datenpaketen und agiert somit unmittelbar oberhalb von Layer 3 des ISO/OSI-Schichtenmodells. Dadurch ist es möglich, dass Daten die durch das Internet geroutet werden, nicht von jedem Mittelsmann entschlüsselt werden müssen, da der IP-Header, der die Routinginformationen enthält, im Klartext vorliegt. IPsec bietet zwei Protokolle und zwei Betriebsmodi, die beliebig miteinander kombiniert werden können.

Bezüglich der Protokolle werden *Authentication Header (AH)* und *Encapsulated Security Payload (ESP)* unterschieden. Während AH ausschließlich Nachrichtenintegrität und -authentizität sicherstellt, bietet ESP darüber hinaus auch eine optionale Verschlüsselung der Daten an. Als Modi kommen Transport- und Tunnelmodus zum Einsatz. Bei ersterem bleibt der IP-Header unverändert und die Mechanismen von AH oder ESP werden nur auf die Payload angewandt. Dieser Modus ermöglicht demnach ausschließlich eine Ende-zu-Ende Verschlüsselung zwischen zwei Parteien, weshalb seine praktische Relevanz eher gering ist ([Eck08]). Im Tunnelmodus hingegen wird das originale IP-Paket vollständig durch AH oder ESP geschützt und ihm wird ein neuer IP-Header – optional mit anderen Sender- und Empfängeradressen – vorangestellt. Damit ermöglicht der Tunnelmodus nicht nur einen Ende-zu-Ende-Schutz, sondern ebenfalls einen Netzwerk-zu-Netzwerk-Schutz. Indem die Security-Gateways zweier Netzwerke Quell- und Ziel-IP der tatsächlich kommunizierenden Teilnehmer durch ihre eigenen austauschen, werden die beiden Netzwerke zu einem *Virtuellen Privaten Netzwerk (VPN)* verbunden. Das hat den Vorteil, dass innerhalb des Netzwerks nur die Gateways IPsec implemen-

tieren müssen. Auch die Verwaltung beschränkt sich in diesem Fall auf die Gateways des Netzwerks. Außerdem ist von außen nur sichtbar, dass Pakete zwischen den Netzwerken verschickt werden, es werden jedoch keine Informationen über die innere Struktur der Netzwerke preisgegeben. [Eck08, SSP08]

Bereits im Jahr 2000 äußerten FERGUSON und SCHNEIER in [FS00] deutliche Kritik an IPsec. Trotzdem sie herausstellten, dass IPsec zweifellos die derzeit beste Möglichkeit ist, IP-Verkehr zu sichern, kritisieren sie, dass die IPsec-Spezifikationssuite schlicht zu komplex sei. Aufgrund dieser Komplexität halten die Autoren es für sehr wahrscheinlich, dass einerseits die Spezifikation mögliche Angriffe erlaubt, andererseits aber auch Implementierungen sehr anfällig für Fehler sind. Zu diesem Schluss kommt auch ECKERT in [Eck08] und weitet diesen Umstand darauf aus, dass die Konfiguration der bestehenden Lösungen ebenfalls sehr kompliziert ist. So ist ein sehr tiefgehendes Verständnis notwendig, um die Wechselwirkungen der verschiedenen Modi und Protokolle abschätzen zu können. Außerdem resultiere die Komplexität in teilweise nicht interoperablen Lösungen verschiedener Hersteller. Weiterhin kritisiert sie, dass Teile der Verwaltung nicht durch das Protokoll selbst, sondern durch die Anwendungsschicht erledigt werden müssen, wodurch das ISO/OSI Schichtenmodell aufgeweicht wird und wiederum schwer abschätzbare Wechselwirkungen entstehen können. Schließlich werden in IPsec MACs eingesetzt, die zwar die Integrität der Nachrichten sichern, jedoch nicht die Eigenschaft der Verbindlichkeit garantieren.

TLS/SSL/DTLS

Das Protokoll *Secure Sockets Layer (SSL)* wurde ursprünglich vom Unternehmen Netscape zum Aufbau gesicherter – also authentifizierter, verschlüsselter und integerer – HTTP-Verbindungen entwickelt, weshalb es oberhalb von Layer 4 im ISO/OSI-Schichtenmodell angesiedelt ist. Nach dem TCP/IP-Referenzmodell befindet sich TLS damit direkt unterhalb der Anwendungsschicht, auf der HTTP angesiedelt ist. In Version 3 war es bereits ein de-facto Standard zum Aufbau sicherer HTTPS-Verbindungen und wurde als *Transport Layer Security (TLS) 1.0* in RFC 2246 veröffentlicht. Die weiteren Entwicklungen erfolgen seitdem an TLS, nicht mehr an SSL, weshalb die beiden Begriffe heute synonym verwendet werden. TLS gehört zu den hybriden Verfahren, es

wird also mittels einer PKI mit asymmetrischen Verfahren eine Verbindung aufgebaut, über die Schlüssel ausgetauscht werden, die für eine symmetrische Verschlüsselung der weiteren Nachrichten genutzt werden. TLS besteht im Wesentlichen aus zwei Teilen, dem *Handshake-Protokoll* und dem *Record-Protokoll*.

Das Handshake-Protokoll dient dem Sitzungsaufbau. Innerhalb eines mehrstufigen Handshakes werden Zertifikate getauscht, um die Parteien zu authentifizieren. Hierbei ist jede Authentifizierungskonfiguration möglich. Insbesondere im Internet – z.B. beim Online-Banking – ist es üblich, nur den Server auf diese Weise zu authentifizieren, da Clients üblicherweise keine eigenen, signierten Zertifikate besitzen. Die Authentifizierung des Clients erfolgt auf höherer Ebene z.B. durch Kombinationen aus Nutzername, PIN und TANs. Denkbar ist dennoch, dass sich nur der Client, Client und Server oder keine der Parteien authentifiziert. Die praktische Relevanz hängt hier von der jeweiligen Anwendung ab. Neben der Authentifizierung dient das Handshake-Protokoll der Einigung auf einen gemeinsamen Schlüssel. Hierbei tauschen die beiden Parteien Nonces, Timestamps und einen Pre-Master-Key aus, aus dem auf beiden Seiten nach einer vorgegebenen Bildungsvorschrift das eigentliche *Master Secret* berechnet wird.

Nach erfolgreicher Durchführung des Handshake-Protokolls ist es Aufgabe des Record-Protokolls, die Anwendungsdaten zu verschlüsseln und ihre Integrität mittels eines MACs zu gewährleisten. Ein SSL-Record bildet dann die Payload eines TCP-Segments. SSL verwendet nicht direkt das Master-Secret, sondern leitet jeweils für Server und Client einen Schlüssel zur Bildung des MAC und einen Schlüssel zur Verschlüsselung ab (vgl. KDFs in Abschnitt 2.2.2). ([Eck08, SSP08]) Auf diese Weise wird direkt unterhalb der Anwendungsebene eine sicherer Kanal geschaffen.

TLS erfordert zwingend die Nutzung von TCP als Transportprotokoll, mit alternativen Protokollen wie UDP lässt es sich also nicht einsetzen. Abhängig von der Anwendung bietet der Einsatz von UDP aber signifikante Vorteile gegenüber TCP, insbesondere hinsichtlich des Protokoll-Overheads. Vor allem in Anwendungen, in denen ein Paket »pünktlich oder gar nicht« eintreffen muss, also Online-Multiplayer-Spiele, Streaming-Anwendungen oder der periodische Versand nicht kritischer Sensorwerte, wird UDP bevorzugt. Da TLS

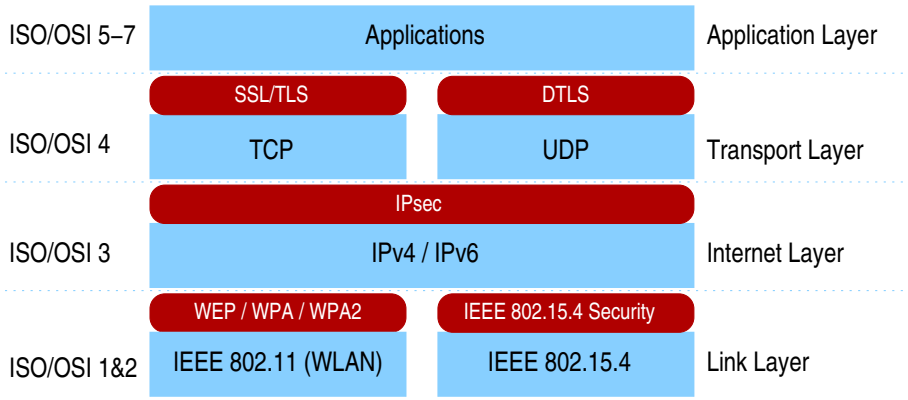


Abbildung 2.5: Einordnung der Sicherheitsmaßnahmen in das TCP/IP- bzw. ISO/OSI-Referenzmodell

weitreichend als Sicherheitsprotokoll akzeptiert ist, spezifizierten Rescorla und Modadugu in RFC 4347 das Protokoll *Datagram Transport Layer Security (DTLS)* ([RM06]). DTLS ist praktisch identisch mit TLS. Der Unterschied liegt darin, dass TLS darauf angewiesen ist, dass Pakete zuverlässig und in der richtigen Reihenfolge übertragen werden, was zwar durch TCP gewährleistet wird, nicht jedoch durch UDP. DTLS stellt diese Eigenschaft während des Handshake-Protokolls wieder her, entfernt die Notwendigkeit jedoch im Rahmen des Record-Protokolls um eine sichere, verbindungslose Kommunikation mit all ihren Eigenschaften zu ermöglichen.

Zusammenfassung und Bewertung

Abbildung 2.5 fasst die bisher beschriebenen Sicherheitsmechanismen zusammen und ordnet sie in die Schichten des TCP/IP Referenzmodells ein. Auf Ebene des Link Layers wurden Mechanismen zur Verschlüsselung von WLAN und zur Sicherung von Funkverkehr über IEEE 802.15.4-konforme Schnittstellen beschrieben. WEP, WPA und WPA2 bieten zwar Schutz vor Abhören der Pakete unberechtigter Dritter, nicht jedoch durch legitime Teilnehmer des Netzwerks. Zudem bietet es nur einen durchgängigen Schutz innerhalb eines Subnetzes und lässt sich nicht routen oder bridgen.

Anders verhält es sich bei IPsec und TLS. Beide haben zum Ziel, Anwendungen sichere Kanäle zwischen zwei Endpunkten und im Fall von IPsec im

Tunnelmodus auch zwischen zwei Netzwerken bereitzustellen. Beide Ansätze haben ihre Vor- und Nachteile, die in Tabelle 2.1 zusammengefasst sind.

	IPsec	(D)TLS
Unabhängigkeit des Transportprotokolls	+	-
Gemeinsame Nutzung eines Tunnels	+	-
Komplexität	-	+
Einfach zu konfigurieren	-	+
Interoperabilität	-	+
Verbreitung	-	+
Vollständig eigenständige Verwaltung	-	+
Performance	o	o
Verbindlichkeit	-	-

Tabelle 2.1: Vor- und Nachteile von IPsec und TLS

Der Einsatz von IPsec bildet keine Abhängigkeit für Transportprotokolle. Anders liegt der Fall bei TLS, das TCP benötigt. DTLS liefert zwar eine Möglichkeit für sichere, verbindungslose Kommunikation via UDP, soll jedoch ein ganz anderes Transportprotokoll zum Einsatz kommen, versagen beide Lösungen. Durch den Tunnelmodus von IPsec können sichere Kanäle zwischen zwei Netzwerken aufgebaut werden, die von mehreren Teilnehmern genutzt werden können. Somit müssen nur die Gateways konfiguriert werden und nicht jeder Client einzeln, wodurch insbesondere Verwaltungs- und Konfigurationsaufwand eingespart wird.

Das hohe Maß an Komplexität von IPsec führt dazu, dass Endpunkte schwer zu konfigurieren und teilweise nicht interoperabel sind, wenn sie von verschiedenen Herstellern stammen. Das Protokoll von TLS ist zwar weniger flexibel als IPsec, dafür jedoch deutlich übersichtlicher, was zu einer weitaus größeren Verbreitung geführt hat. Außerdem kann TLS die Verwaltung der Authentifizierung und der Schlüsselverwaltung vollständig übernehmen, wodurch das Ebenenmodell eingehalten und nicht aufgeweicht wird.

Alshamsi und Saito haben in [AS05] einen Vergleich zwischen IPsec und TLS insbesondere hinsichtlich ihrer Performance angestellt. Darin zeigen sie, dass es keine wesentlichen Unterschiede gibt, die für eine der beiden Techno-

logien sprechen würden. Einzig der Verbindungsaufbau dauert bei IPsec unter Umständen länger als bei TLS. IPsec und TLS teilen auch einen Nachteil, nämlich die mangelnde Verbindlichkeit. Da beide Protokolle MACs einsetzen und keine Signaturen, ist zwar die Integrität der Nachrichten gewährleistet, nicht jedoch ihre Verbindlichkeit ([Eck08]).

Alle beschriebenen Verfahren haben neben ihrer weitestgehenden Untauglichkeit für Kleinstgeräte den gemeinsamen Nachteil, dass sie an eine Übertragungstechnologie gebunden sind (vgl. Abb. 2.5). Die Auswahl einer dieser Technologien bedeutet demnach die Festlegung einer Übertragungstechnologie, wie z.B. WLAN, IP oder TCP. Auch, wenn IP im Internet der Dinge, wie in Abschnitt 2.1 beschrieben, als vielversprechender Kandidat gilt, so muss berücksichtigt werden, dass die Anbindung von z.B. Bluetooth-Komponenten in intelligenten Umgebungen Teil der meisten gängigen Szenarien ist. Technologien wie Bluetooth fügen sich nicht in das ISO/OSI-Layermodell ein, sondern definieren vollständige Netzwerk-Stacks, auf denen Anwendungen aufsetzen können. Eine Sicherheitsinfrastruktur für intelligente Umgebungen muss demnach auf Anwendungsebene implementiert werden, um protokollübergreifenden Schutz bieten zu können. Existierende Mechanismen auf Anwendungsebene werden gemeinsamen mit verschiedenen Kommunikationstechnologien für intelligente Umgebungen in Kapitel 6.1 diskutiert.

2.3 Service-orientierte Architekturen, Web Services und DPWS

Serviceorientierte Architekturen (SOA) im Allgemeinen und Web Services (WS) im Speziellen erfreuen sich seit Mitte der Neunziger steigender Beliebtheit als Werkzeug zur Kommunikation in Geschäftsprozessen. Der Grund hierfür liegt darin, dass sich Geschäftsprozesse sehr einfach auf Web Services abbilden lassen, da sie einzelne Prozessfunktionalitäten als Dienste kapseln. Ein Designziel von SOAs ist die lose Kopplung einzelner Dienste. Durch definierte Schnittstellen lassen sich so einzelne Dienste austauschen, ohne dass das restliche System dies berücksichtigen muss ([NPS08, FZ09]). Web Services nutzen *SOAP* als Basisprotokoll, dass seinerseits in XML abgebildet wird. Eine SOAP-Nachricht wird als *SOAP-Envelope* bezeichnet, der in einen optionalen

SOAP-Header für Metainformationen und den *SOAP-Body* für die Nutzlast unterteilt.

Um Geräte wie z.B. Drucker oder Scanner ebenso wie alle übrigen Ressourcen in geschäftliche Infrastrukturen und damit verbunden nahtlos in bestehende oder neue Prozesse einbinden zu können, begann 2004 die Arbeit am *Devices Profile for Web Services (DPWS)*. Maßgeblich von Microsoft und IBM unterstützt, wurde DPWS 2006 ([Mic06]) in Version 1.0 spezifiziert. Die bis heute aktuelle Version 1.1, an der auch das Institut für Angewandte Mikroelektronik und Datentechnik der Universität Rostock beteiligt war, erschien 2009 ([OAS09a]). DPWS besteht aus einer Untermenge, einem sogenannten Profil, der verschiedenen Spezifikationen des umfangreichen Web-Service-Universums. Diese Untermenge bietet zunächst einen kompakten Weg, Web-Services, die auf Geräten gehostet werden, zu adressieren und anzusprechen. Ergänzt wird DPWS durch Mechanismen zur Beschreibung von Geräten und gehosteten Diensten ([BB+06]), einer Möglichkeit für ein dynamisches Discovery von Geräten ohne zentrale Registry ([OAS09b]) und eines asynchronen Publish-Subscribe-Mechanismus ([W3C06b]).

Obwohl es für Geräte wie Drucker und Scanner im Unternehmenseinsatz entworfen wurde, existieren Strömungen, die DPWS im Umfeld drahtloser Sensornetze ([MZP+09]), im industriellen Kontext ([MPTG08]) und zur Heimautomatisierung ([Exc11]) einsetzen. Es stellt weiterhin die Basis für eine künftige Technologie zur Vernetzung von medizinischen Geräten im Klinikalltag dar [KKU+15]. Das Sicherheitskonzept von DPWS basiert auf dem Konzept frei wählbarer Sicherheitsprofile. Die Basisspezifikation schlägt ein solches Profil vor und verwendet dabei TLS und einen kompakten Signaturmechanismus. Eine detaillierte Besprechung und Analyse erfolgt in Kapitel 6.1.1.

Wegen seiner Vielseitigkeit und der Tatsache, dass DPWS vollständig auf offenen, weit verbreiteten und lizenzfreien Internettechnologien basiert, wird es im Verlauf der gesamten Arbeit als Basisprotokoll für eine dienstbasierte Kommunikationstechnologie angenommen.

2.4 Die Web Service Security Spezifikationssuite

Mit der wachsenden Beliebtheit und dem daraus resultierenden ansteigenden Einsatz von Web Services in Unternehmensstrukturen stellten sich Fragen nach der Kommunikationssicherheit. Insbesondere vor dem Hintergrund, dass eine Orchestrierung von Diensten auch beinhalten kann, dass Dienste verschiedener Unternehmen zusammenarbeiten sollen, um ein gemeinsames Ziel zu erreichen, müssen grundlegende Sicherheitsmechanismen realisiert werden. Daher wurde eine Reihe zusätzlicher, sicherheitsorientierter WS-Spezifikationen geschaffen. Nachfolgend werden die zum Zeitpunkt der Erstellung dieser Arbeit bestehenden Spezifikationen beschrieben. Das Zusammenspiel der Funktionalitäten der einzelnen Spezifikationen wird in Abschnitt 2.4.7 erläutert.

2.4.1 WS Security

WS Security [OAS06c] bietet zunächst die grundlegenden Funktionen zur Sicherung von SOAP-Nachrichten an und definiert hierzu zunächst die entsprechenden Header-Formate. Innerhalb dieser Spezifikation wird erläutert, wie Teile einer Nachricht verschlüsselt und/oder kryptografisch signiert werden können, wobei die entstehende Chiffre bzw. die Signatur in die SOAP-Nachricht eingebettet werden. Außerdem wird ein Format für Zeitstempel in SOAP-Nachrichten definiert, um den Zeitpunkt der Erstellung einer Nachricht festzuhalten. WS Security bedient sich dabei zweier weiterer Basisspezifikationen. XML-Signature ([BBF+08]) definiert, wie die kryptografische Signatur eines SOAP-Envelopes gebildet und anschließend in selbigen eingebettet wird. XML-Encryption ([IDS02]) beschreibt, wie Teile einer SOAP-Nachricht verschlüsselt und die entstehende Chiffre anschließend mit der Nachricht transportiert werden kann. Da diese beiden Basisspezifikationen einen wichtigen Aspekt in Kapitel 3 darstellen, werden sie dort besprochen.

Schließlich werden für fast alle kryptografischen Routinen zusätzliche Daten benötigt, wie z.B. die X.509-Zertifikate der Teilnehmer, welche deren öffentliche Schlüssel enthalten. Derartige zusätzliche Daten werden in der Spezifikation WS Security als sogenannte *Security Tokens* bezeichnet und es wird definiert, wie ein Token in eine Nachricht eingebunden oder in ihr referenziert werden kann. Zu jeder Art unterstützter Security Tokens existiert ein *Security Token*

Profile, das den Aufbau des jeweiligen Tokens regelt. So lassen sich X.509-Zertifikate [OAS06f] oder Nutzername-Passwort-Kombinationen [OAS06e] in einer SOAP-Nachricht übertragen. Weitere Tokens beziehen sich auf den Transport von Kerberos-Tokens (vgl. Abschnitt 2.2.2, [OAS06a]) und Beschreibungssprachen wie die Rights Expression Language (REL) ([OAS04, Int04] und die Security Assertion Markup Language (SAML, [CKPM05, OAS06b]). Alle weiteren Daten lassen sich mit dem WSS SOAP with Attachments Profile ([OAS06d, W3C00]) sicher übertragen.

2.4.2 WS Secure Conversation

Wie eine SOAP-Nachricht verschlüsselt und kryptographisch signiert werden kann, wird durch die Spezifikation WS Security (vgl. Abschnitt 2.4.1) geregelt. Der Standard WS Secure Conversation ([OAS07a]) nutzt diese Funktionalität, um einen sicheren Kontext zwischen zwei Kommunikationsparteien aufzubauen. Ein Kontext bedeutet hier eine Verbindung zwischen zwei Parteien innerhalb derer mehrere aufeinanderfolgende Nachrichten ausgetauscht werden.

Ein sicherer Kontext wird in WS Secure Conversation durch ein *Security Context Token (SCT)* abgebildet. Ein solches Token kann durch eine zentrale vertrauenswürdige Instanz im Netzwerk – einem sogenannten Security Token Service (vgl. Abschnitt 2.4.3) – ausgestellt werden. Ist eine solche Instanz nicht vorhanden (oder ihre Teilnahme nicht erwünscht), kann einer der beiden Kommunikationsteilnehmer ein Security Context Token ausstellen oder es kann zwischen beiden Teilnehmern ausgehandelt werden.

2.4.3 WS Trust

Im Zusammenhang mit WS Security wurden in Abschnitt 2.4.1 auch die Security Tokens erläutert, mit denen kryptografisch relevante Informationen wie z.B. Zertifikate ausgetauscht werden können. Während WS Security beschreibt, wie diese Tokens in Nachrichten eingebettet werden können, beschreibt die Spezifikation WS Trust [OAS06g] das notwendige Nachrichten-Framework, um solche Tokens anzufragen, auszuliefern, sie zu erneuern oder zurückzurufen. Einerseits kann dies zwischen zwei Diensten direkt geschehen. Zusätzlich definiert WS Trust einen speziellen Service innerhalb eines Netzwerks, den *Security*

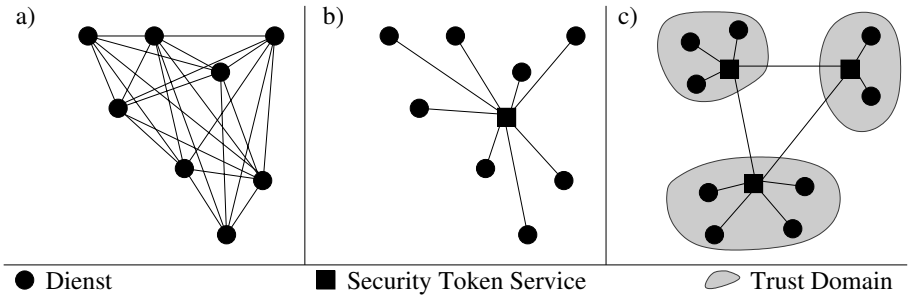


Abbildung 2.6: Verschiedene Vertrauensbeziehungen
a) ohne Security Token Service b) mit Security Token Service
c) mit Security Token Service zwischen verschiedenen Trust Domains

Token Service (STS). Hierbei wird davon ausgegangen, dass jeder Teilnehmer eines Netzwerks dem Security Token Service vertraut. Dadurch können Clients Token – wie zum Beispiel ein SCT – vom STS anfordern und diese beim Aufruf eines Dienstes mitliefern. Der Dienst selbst kann dann entweder das vom STS ausgestellte Token selbst verifizieren oder es an den STS zur Überprüfung übergeben. Dadurch müssen Vertrauensbeziehungen nicht zwischen allen Teilnehmern, sondern immer nur zum Security Token Service aufgebaut werden (vgl. Abb. 2.6 a) und b)). Anschließend ist eine delegierte Authentifizierung der übrigen Teilnehmer möglich.

Der Einsatz von Security Token Services erleichtert außerdem die sichere Kommunikation zwischen Diensten in verschiedenen Netzwerken, die verschiedene Trust Domains abbilden. Dabei müssen Vertrauensbeziehungen immer nur zwischen den Teilnehmern eines Netzwerks und ihrem STS und unterhalb der STS selbst hergestellt werden (vgl. Abb. 2.6 c)).

Schließlich definiert WS Trust das notwendige Nachrichten-Framework, um Aushandlungen zwischen zwei Teilnehmern zu ermöglichen. Dies kann z.B. zum Einsatz kommen, wenn die Parameter eines sicheren Kontextes ausgehandelt werden sollen wie in Abschnitt 2.4.2 beschrieben.

2.4.4 WS Federation

Zuvor wurde in Abschnitt 2.4.3 erläutert, dass WS Trust den Aufbau von Vertrauensbeziehungen zwischen verschiedenen Trust-Domains, z.B. mehrerer Unternehmen, vereinfacht. Auf diese Möglichkeit setzt die Spezifikation WS Fe-

deration ([L+06]) auf. Eine *Federation* ist ein Bündnis kooperierender Unternehmen oder Institutionen wobei WS Federation zum Ziel hat, die Sicherheit der Kommunikation zwischen Angehörigen dieser Institutionen zu erhöhen und gleichzeitig kosteneffizienter zu gestalten. Erreicht wird dieses Ziel, in dem jeder Teilnehmer einer Federation die Sicherheitskonfiguration seiner eigenen Angehörigen übernimmt und die daraus resultierenden Informationen mit seinen verbündeten Institutionen teilt.

Während WS Trust das Nachrichten-Framework bereitstellt, um Dienste und Clients über die Grenzen einer Trust Domain hinweg zu authentifizieren, definiert WS Federation konkrete Authentifizierungsmethoden, die genutzt werden können. Ist ein Client authentifiziert, muss weiterhin sichergestellt sein, dass er die entsprechende Autorisierung besitzt, um einen Dienst anzusprechen. Hierzu definiert WS Federation eine spezielle Form des Security Token Service (vgl. Abschnitt 2.4.3), den *Authorization Service*. Zu diesem Zweck wird gleichzeitig das Nachrichten-Framework von WS Trust erweitert, um Autorisierungsanfragen mit zusätzlichen Informationen zum Anfragesteller zu versehen. Da verschiedene Institutionen häufig verschiedene Autorisierungsprotokolle befolgen, definiert WS Federation ein protokollunspezifisches Nachrichten-Framework, um die Interoperabilität zwischen den Institutionen zu vereinfachen.

Um den Austausch von Informationen zu Anfragestellern und Diensten zu vereinfachen, bietet WS Federation mehrere Möglichkeiten. Mit Hilfe des Metadaten-Frameworks können geschäftsbezogene Zusatzinformationen zu einzelnen Diensten zugänglich gemacht werden, die nicht durch die WSDL-Beschreibung des Dienstes oder dessen Policies (vgl. Abschnitt 2.4.5) ausdrückbar sind. Um anzuzeigen, welche Formen der Verschlüsselung akzeptabel sind, wird die Syntax der WS Trust-Nachrichten erweitert. Gleichzeitig wird eine Sonderform der Fehlermeldung eingeführt, falls diese Vertraulichkeitsanforderungen nicht eingehalten werden. Diese Fehlermeldung wird um Angaben zu den unterstützten bzw. erforderlichen Verfahren angereichert. Schließlich wird ein *Attribute Service* definiert. Dieser soll es den anderen Teilnehmern einer Federation ermöglichen, Informationen über einen Teilnehmer innerhalb der eigenen Trust-Domain einzuholen.

Da sich auch die Konfiguration der Teilnehmer einer Federation dynamisch ändern kann, wird der *Federated Sign-Out*-Mechanismus definiert. Über ihn kann dem gesamten Bündnis mitgeteilt werden, dass ein Teilnehmer die Federation verlassen hat, um z.B. seine Berechtigungen aus den Datenbanken zu löschen. Um den Missbrauch von Identitäten zu erschweren, kann schließlich ein Pseudonymisierungsdienst eingeführt werden, der sich nahtlos in einen Security Token Service integrieren lässt. Für die einzelnen Teilnehmer ist dieser Dienst völlig transparent, sodass kein zusätzlicher Aufwand betrieben werden muss ([GHN+07]).

2.4.5 Policies

Insbesondere im Sicherheitsbereich müssen Dienste in der Lage sein, bestimmte Parameter, wie z.B. unterstützte kryptografische Algorithmen oder zulässige Formen der Authentifizierung bekannt zu geben. Die Spezifikation WS Policy [W3C07] definiert ein allgemeines Framework zur Beschreibung solcher Parameter und die Spezifikation WS Security Policy[OAS07b] definiert konkrete sicherheitsbezogene Policies. Abschließend beschreibt WS-PolicyAttachment [W3C06a], wie diese Policies in die WSDL-Beschreibungen der Dienste eingebettet werden können.

2.4.6 Security Assertion Markup Language (SAML)

In Abschnitt 2.4.4 wurde das Konzept von Federations beschrieben, in denen Entitäten verschiedener Sicherheitsdomänen zusammenarbeiten können. Die *Security Assertion Markup Language* (SAML, [RHP+08]) bietet in Version 2.0 einen portablen, interoperablen Weg, Authentifizierungs- und Autorisierungsanfragen sowie Informationen über Entitäten innerhalb von Federations auszutauschen. Um Konzepte wie Single-Sign-On und verteilte Transaktionen zu ermöglichen, können über den in WS Trust (vgl. Abschnitt 2.4.3) definierten STS SAML-Tokens beantragt werden, die sogenannte *Assertions* beinhalten. Diese Assertions beinhalten *Statements* z.B. zu einer föderierten Authentifizierung oder Autorisierung. [Bun09]

2.4.7 Zusammenfassung

Um das Zusammenspiel und die Einsatzbereiche der einzelnen Spezifikationen zu verdeutlichen, werden sie an dieser Stelle noch einmal gemeinsam betrachtet. Dabei zeigt sich, dass jede Spezifikation die sicherheitsbezogenen Probleme einer Abstraktionsebene beim Austausch von SOAP-Nachrichten löst. Aus diesem Grund ist der übrige Abschnitt in genau diese Abstraktionsebenen gegliedert und in ihrem Zusammenhang werden die jeweiligen Spezifikationen beschrieben. Abbildung 2.7 fasst das Ebenenmodell der Web Service Security Spezifikationen zusammen.

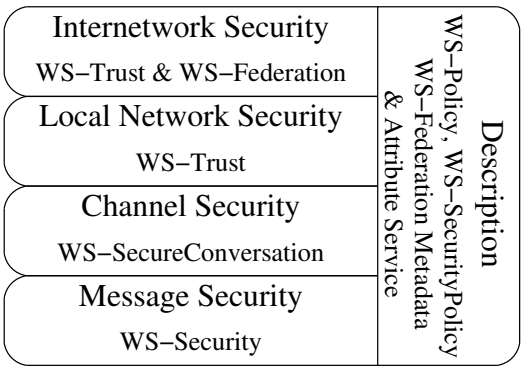


Abbildung 2.7: Ebenenmodell der Web Service Security Spezifikationen

Nachrichtenebene

Das atomare Problem in einem Netzwerk, in dem SOAP-Nachrichten gesichert ausgetauscht werden sollen, ist die Sicherung einer SOAP-Nachricht selbst. Lösungen für dieses Problem bietet die Spezifikation WS Security, in dem sie definiert, wie *eine Nachricht* gesichert werden kann (vgl. Abbildung 2.8 a)). Hierzu zählt das digitale Signieren und Einbetten verschlüsselter Nachrichtenteile ebenso wie das Einbetten von zusätzlichen Informationen wie Zeitstempel oder Zertifikate.

Verbindungsebene

Die erste Abstraktionsebene oberhalb einer einzelnen Nachricht ist eine *Verbindung* innerhalb derer zwischen zwei Teilnehmern *mehrere Nachrichten* gesichert ausgetauscht werden. Zu diesem Zweck definiert WS Secure Conver-

sation eine Möglichkeit, *eine sichere Verbindung zwischen zwei Teilnehmern* effizienter zu gestalten, als wenn jede Nachricht isoliert betrachtet wird (vgl. Abbildung 2.8 b)).

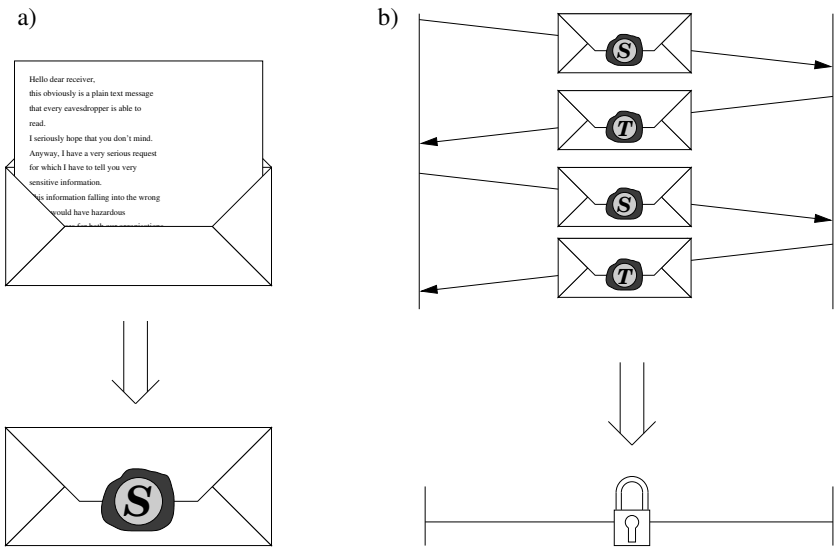


Abbildung 2.8: Zusammenspiel der WS Security Spezifikationen
a) WS Security auf Nachrichtenebene
b) WS Secure Conversation auf Verbindungsebene

Lokale Netzwerkebene

Der größte Vorteil service-orientierter Architekturen liegt in der losen Kopplung *vieler* Komponenten in einem Netzwerk. Demnach ist die nächsthöhere Abstraktionsebene nach einer Verbindung zwischen zwei Teilnehmern das *gesamte Netzwerk* einer Administrationsdomäne. Hierzu führt WS Trust den Security Token Service ein, der eine zentrale vertrauenswürdige Autorität innerhalb eines Netzwerks darstellt. Seine Aufgabe ist das Ausstellen, Verifizieren und Verwalten von den durch WS Security definierten Security Tokens. So lassen sich auch möglichst effizient Vertrauensbeziehungen zwischen allen Teilnehmern des Netzwerks aufbauen. Indem Vertrauensbeziehungen aller Teilnehmer mit dem Security Token Service hergestellt werden, vertrauen sich auch Teilnehmer untereinander implizit (vgl. Abbildung 2.6 a) und b)).

Internetzwerkebene

WS Trust berücksichtigt weiterhin die Kooperation von Teilnehmern innerhalb verschiedener administrativer Domänen in dem es den Austausch sicherheitsrelevanter Informationen durch den Einsatz von Security Token Services vereinfacht. Darauf aufbauend ermöglicht WS Federation den Austausch von Authentifizierungs- und Autorisierungsinformationen zwischen den Domänen.

Beschreibungsebene

Unabhängig von den verschiedenen Netzwerkebenen müssen Möglichkeiten bestehen, die Teilnehmer eines Netzwerks zu beschreiben, um qualifizierte Sicherheitsentscheidungen treffen zu können. Das gilt sowohl für Dienste als auch für deren Konsumenten. Für Dienste steht hier zunächst deren WSDL-Beschreibung zur Verfügung, in die mittels WS Security Policy sicherheitsrelevante Informationen eingebettet werden können. Um Konsumenten zu beschreiben, definiert WS Federation den Attribute Service.

Kapitel 3

Performance von Sicherheit auf Netzwerk- und Anwendungsebene

Die in diesem Kapitel beschriebenen Erkenntnisse und Vorschläge für WS Compact Security und die WS Security Records wurden im Rahmen der Anfertigung dieser Dissertation bereits als folgende Publikation veröffentlicht:

> *Sebastian Unger, Stefan Pfeiffer, Dirk Timmermann „Dethroning Transport Layer Security in the Embedded World“ in 5th International Conference on New Technologies, Mobility and Security (NTMS), 2012. DOI: 10.1109/NTMS.2012.6208685 [UPT12a]*

In diesem Kapitel wird eine Alternative zur häufig verwendeten Protokollgruppe *Transport Layer Security* (TLS) für Geräte vorgestellt, um die Machbarkeit und grundsätzliche Anwendbarkeit eines Sicherheitskonzepts auf Anwendungsebene nachzuweisen. Zunächst werden die Nachteile von TLS zusammengefasst. Anschließend wird ein Lösungsvorschlag skizziert und dessen erwartete Ergebnisse genannt. Es folgt eine detaillierte Beschreibung der entwickelten Lösung samt Performance-Analyse und einer Diskussion.

Um die Lesbarkeit dieses Kapitels zu erhöhen, wurden Messreihen und Beispielnachrichten in Anhang A.1 ausgelagert. Die spezifizierten Nachrichtenformate sind detailliert in Anhang B.7 dargestellt.

3.1 Einführende Überlegungen

In Abschnitt 2.2.3 wurde die Protokollgruppe Transport Layer Security (TLS) besprochen. TLS wird häufig als Sicherheitsinfrastrukturkomponente eingesetzt, so auch beim Entwurf neuer Kommunikationstechnologien für eingebettete Systeme. Das hat verschiedene Gründe. So ist TLS eine Technologie, die bereits 1999 eingeführt wurde ([DA99]), ist also viele Jahren gereift, bevor sie ihre heutige Form ([DR08]) annahm. Sein Vorgänger, das Protokoll Secure Sockets Layer (SSL) wurde vom Unternehmen Netscape entwickelt ([SSP08]) und war bereits 1994 im Browser MOSAIC zu den Anfängen des World Wide Web enthalten, was die Reife dieser Protokollgruppe weiter erhöht.

Zudem ist TLS insbesondere im Internet sehr weit verbreitet und stellt das Mittel der Wahl beim Aufbau gesicherter Kanäle zwischen zwei Parteien über ungesicherte Netzwerke wie dem World Wide Web dar. Das betrifft nicht nur die Verschlüsselung privater Kommunikation wie etwa beim Online-Banking, sondern auch die Absicherung sensibler geschäftlicher oder behördlicher Kommunikationsprozesse.

Die Verbreitung von TLS und das Vertrauen, das in diese Protokollgruppe gesetzt wird, geht so weit, dass ein steigender Teil virtueller privater Netzwerke (VPNs) durch TLS abgesichert wird ([BMC+05]), obwohl der Tunnel-Modus von IPSec für genau diesen Zweck entwickelt wurde (vgl. 2.2.3).

TLS stellt demnach eine weit verbreitete Möglichkeit dar, Netzwerkverkehr abzusichern, die sich zudem über Jahrzehnte behaupten konnte. Ein weiteres entscheidendes Kriterium für seine Verwendung auch in eingebetteten Systemen ist schließlich seine vergleichsweise einfache Verwendung. TLS steht in gleich mehreren offenen Bibliotheken zur Verfügung (z.B. [Ope14], [MJ14] oder [Leg14]) und kann so einfach in abzusichernde Applikationen eingebunden werden. Da es sich im ISO-OSI-Layer-Modell unmittelbar über Schicht 4 befindet (bzw. zwischen Application- und Transport Layer im TCP/IP-Referenzmodell, vgl. Abbildung 2.5), ist es für die darauf aufbauende Anwendung vollständig transparent.

3.1.1 Probleme mit dem Stand der Technik

Während TLS die bevorzugte Methode ist, Netzwerkverkehr in zentralisierten Client-Server-Umgebungen abzusichern, birgt es Nachteile bei der Verwendung durch dezentral organisierte Systeme, die eine hohe Zahl ressourcenschwacher Geräte beinhalten ([MLH+08], [HLP+09], [UZG+10], [UPT12b] und [UPT12a]).

Die Authentifizierung basiert bei TLS in seiner heutigen Form zwingend auf X.509-Zertifikaten ([BMC+05, cpt.7.3.5]) (vgl. Kapitel 2.2.2). Die Speichergröße solcher Zertifikate beansprucht ressourcenschwache Geräte jedoch stark, da nicht nur das eigene gespeichert werden muss. Stattdessen findet ein gegenseitiger Austausch mit jedem Authentifizierungspartner statt, was zu großen zu übertragenden und zu persistierenden Datenmengen führt.

Eine weitere Schwierigkeit im Umgang mit Zertifikaten besteht in ihrem sicheren Austausch und damit in der eigentlichen Authentifizierung auf Basis der Zertifikate. In Kapitel 2.2.2 wird beschrieben, wie Public-Key-Infrastrukturen mithilfe von Zertifikaten aufgebaut werden können. Empfängt ein Teilnehmer ein ihm bisher unbekanntes Zertifikat, muss er die Identität der Certificate Authority (CA) überprüfen, die dieses Zertifikat signiert hat. Gelingt dies nicht, muss der Teilnehmer die Hierarchie in Richtung der Wurzel (rootCA) traversieren, bis er eine CA findet, der er vertraut. Dieses Vorgehen erhöht die zu übertragenden und zu speichernden Datenmengen weiter und verlangt außerdem, dass eine herstellerübergreifende Zertifikatsinfrastruktur errichtet wird, die jedoch zentral verwaltet werden muss. Außerdem zeigt die Vergangenheit die Angreifbarkeit solcher Infrastrukturen. Wiederholt wurden Zertifikate von CAs und rootCAs gestohlen, wodurch gefälschte Zertifikate ausgestellt werden konnten und somit die Sicherheit des gesamten Systems kompromittiert wurde [Hig11, Sch15, Kap15].

Alternativ zur Nutzung einer PKI können Zertifikate z.B. einer CA auch direkt auf die Geräte installiert werden. Dies muss allerdings vor der Ausbringung der Geräte ins Feld geschehen (demnach vor der Auslieferung an Kunden oder Händler) und kann nicht durch den Nutzer zur Laufzeit vorgenommen werden. Eine vollständig herstellerübergreifende Lösung ist damit nicht umsetzbar. Außerdem ist es kaum möglich, so auf Änderungen wie z.B. kompromittierte oder abgelaufene Zertifikate zu reagieren.

Die Nachteile bei der Authentifizierung mittels X.509-Zertifikaten sind demnach unmittelbar Nachteile von TLS, da keine alternativen Authentifizierungsmethoden vorgesehen sind. Ein Vorteil von TLS, seine Transparenz für Anwendungen, kann auch ein Nachteil sein, denn die Daten treffen unvermeidlich unverschlüsselt in der Anwendung ein. Wird hingegen nur die Nutzlast auf Anwendungsebene verschlüsselt, so begünstigt das eine sichere Persistenz der Daten, da kein zusätzlicher Aufwand betrieben werden muss, um die Daten verschlüsselt abzulegen. Eine weitere Möglichkeit die sich dadurch ergibt, ist eine Ende-zu-Ende-Verschlüsselung, da Anwendungen Daten weiterleiten können, ohne sie selbst zu entschlüsseln (sog. Multihop) ([Mas09]).

Schließlich ist TLS auf Transportebene an das Transport Control Protokoll (TCP) gebunden. Zwar existieren Adaptionen wie die Protokollgruppe Datagram Transport Layer Security (DTLS, [RM06]), die die Prinzipien von TLS auf alternative Transportprotokolle wie in diesem Fall das User Datagram Protocol (UDP) anwenden (vgl. Kapitel 2.2.2). Es bleibt jedoch die Bindung an ein konkretes Transportprotokoll. Eine sichere Anbindung von Technologien mit eigenen Netzwerk-Stacks wie Bluetooth oder ZigBEE ist nicht möglich.

3.1.2 Lösungsansatz und erwartete Ergebnisse

Die im vorigen Abschnitt genannten Probleme lassen sich durch einen Sicherheitsansatz auf Anwendungsebene lösen. Unterstützt eine entsprechende Lösung alternative Authentifizierungsmethoden (vgl. Kapitel 2.2.2), so werden die Probleme bei der Authentifizierung mittels X.509-Zertifikaten nicht übernommen. Treffen die Daten verschlüsselt auf Anwendungsebene ein, lassen sich auch Multihop-Szenarien mit Ende-zu-Ende-Verschlüsselung realisieren und eine verschlüsselte Persistenz der Daten wird begünstigt. Schließlich ist ein Konzept auf Anwendungsebene vollständig unabhängig vom zugrundeliegenden Netzwerkstack. Eine sichere Interoperabilität mit z.B. Bluetooth oder ZigBEE ist somit möglich.

Bei den zu erwartenden Vorteilen einer Lösung auf Anwendungsebene darf jedoch kein gravierender Nachteil zu TLS als Stand der Technik entstehen, der die Akzeptanz dieser Lösung gefährdet. Da sich TLS nicht zuletzt durch seine Schnelligkeit auszeichnet, darf eine entwickelte Lösung bei gleichem Funktionsumfang demnach keine merklichen Performance-Einbußen mit sich bringen.

Dem folgt die Überlegung, dass bei einer Absicherung mittels TLS der gesamte Transportkanal verschlüsselt wird, auf Anwendungsebene jedoch nur die Nutzlast. Als Beispiel soll die Nachricht in Listing A.1 dienen. Es handelt sich dabei um eine SOAP-Nachricht, die über HTTP übertragen wird. Wird diese mit TLS verschlüsselt, betrifft das die gesamte TCP-Nutzlast, also den HTTP-Header und die SOAP-Nachricht. Dies entspricht einer Summe von 352 Byte. Dem gegenüber steht eine Nutzlast von 107 Byte, die auf Anwendungsebene verschlüsselt werden müsste.

Im genannten Beispiel müsste ein Ansatz auf Anwendungsebene demnach weniger als ein Drittel der Klartextmenge verschlüsseln, die TLS auf Transportebene zu bewältigen hätte. Zur praktischen Durchführbarkeit des Austauschs verschlüsselter Nachrichten fehlen jetzt jedoch Informationen wie z.B. Angaben über den verwendeten Verschlüsselungsalgorithmus oder eine Referenz auf den verwendeten Schlüssel. Außerdem stellt Verschlüsselung alleine die Nachrichtenintegrität nicht sicher, es muss demnach auch eine Signatur der Nachricht eingebettet werden. Um eine SOAP-Nachricht um die benötigten Informationen anzureichern, existieren die Spezifikationen XML-Encryption und XML-Signature. Erstere beschreibt, wie verschlüsselte Nachrichtenteile in ein XML Dokument eingebettet werden und letztere erklärt das Anhängen einer kryptografischen Signatur. Zusammengefasst werden sie durch die Spezifikation WS Security als Mechanismus, die Nutzlast einer einzelnen SOAP-Nachricht zu verschlüsseln und wichtige Dokumentteile zu signieren (vgl. Kapitel 2.4.1). Wird die in Listing A.1 dargestellte SOAP-Nachricht mit diesem Mechanismus geschützt, entsteht die Nachricht in Listing A.2. Aus den vormals 236 Byte der SOAP-Nachricht in Listing A.1 entstehen 1862 Byte. Hierbei handelt es sich um eine Minimalform, ein praktisches Beispiel würde weitere `<ds:SignedInfo-` Elemente enthalten und wäre demnach mehrere hundert Byte größer bei einer Gesamtgröße von ca. 2500 Byte.

Demnach stehen dem stark reduzierten Verschlüsselungsaufwand (Faktor drei) dramatisch vergrößerte Nachrichten gegenüber (beinahe eine Größenordnung). Für verteilte eingebettete Systeme, die oft batteriebetrieben sind und häufig drahtlos kommunizieren, stellt das keine praktikable Lösung dar, da drahtlose Kommunikation der teuerste Posten in der Energiebilanz ist ([RSPS02]). Nachfolgend wird untersucht, in welchem Maße der durch WS Se-

curity eingeführte Overhead der Nachrichtengröße reduziert werden kann. Anschließend wird geprüft, ob die Performance des entwickelten Ansatzes hoch genug ist, um mit TLS als derzeitigem Stand der Technik zu konkurrieren.

3.2 WS Compact Security als kompakte Darstellung von XML Signature und XML Encryption

Wie eingangs erwähnt, setzt sich WS Security maßgeblich aus einem Mechanismus zum Einbetten von Chiffren in ein XML-Dokument (XML-Encryption, [IDS02]) und einem Mechanismus zum Anhängen kryptografischer Signaturen an ein Dokument (XML-Signature, [BBF+08]) zusammen. Um die Kompatibilität zu existierenden Implementierungen von WS Security zu wahren, muss ein zu entwerfendes WS Compact Security entsprechend aus je einem kompakten Mechanismus zum Anhängen von Signaturen und zur Einbettung von Chiffren bestehen.

Im Folgenden wird zunächst das bereits durch [OAS09b] spezifizierte kompakte Signaturformat WSDD Compact Signatures besprochen und mit klassischen XML-Signatures verglichen. In Anlehnung an die existierenden WS Compact Signatures wird ein kompaktes Schema zur Einbettung von Chiffren vorgeschlagen und beides zu dem kompakten Schema WS Compact Security kombiniert. Abschließend erfolgt ein Leistungsvergleich mit TLS als Stand der Technik.

3.2.1 Kompaktes Anhängen von Signaturen

Um die Größenreduktion durch die Verwendung des kompakten Schemas zu verdeutlichen, vergleicht Abbildung 3.1 die schematischen Darstellungen der Signatur einer SOAP-Nachricht, wobei vier Elemente dieser Nachricht in die Signaturbildung eingehen. Die linke Seite stellt die angehängte Signatur nach dem XML-Signature-Schema dar, die rechte Seite nach dem WSDD Compact Signature Schema. Die Zuordnung der einzelnen Angaben in beiden Schemata wurde farblich markiert. Die Nachrichten sind detailliert in den Listings A.3 und A.4 dargestellt.

Abbildung 3.1: Vergleich von XML-Signature (links) und WSDD Compact Signature (rechts)

Um eine XML-Signature zu bilden, wird zunächst für jedes zu signierende Element einer Nachricht ein `<SignedInfo>`-Block angelegt (gelb dargestellt). In diesem Beispiel sind das die Elemente `<RelatesTo>`, `<Action>`, `<Timestamp>` und `<Body>`. Für jedes dieser Elemente enthält der `<SignedInfo>`-Block die Kanonisierungsmethode, eine Angabe, mit welchem Algorithmus der Digest des Elements zu einer Signatur verschlüsselt wurde und eine Referenz auf das zu signierende Element selbst (violett). Die Referenz selbst enthält zunächst Angaben dazu, mit welchem Algorithmus der Digest des Elements gebildet wurde und den Digest selbst. Schließlich enthält das `<Transforms>`-Element eine Liste von Verarbeitungsschritten, die vor der Bildung des Digests auf das Element angewendet wurden. Typisch ist auch hier eine Kanonisierung. Hierzu wird schließlich angegeben, welche Namespaces dabei berücksichtigt wurden (magenta hinterlegt).

All diese `<SignedInfo>`-Blöcke werden zusammen mit dem Wert der Signatur (blau) und Angaben über den verwendeten Schlüssel (braun) in ein `<Signature>`-Element (grün) eingefügt.

All diese Informationen lassen sich verlustfrei als eine WSDD-Compact-Signature (Abb. 3.1 rechts) abbilden, indem kompaktere Darstellungsformen gewählt werden und folgende Annahmen getroffen werden. Zunächst wird festgelegt, dass jedes Element mit der C14N-Methode ([BER02]) kanonisiert wird.

Außerdem wird festgelegt, dass alle Elemente mit dem selben Satz an Algorithmen kanonisiert, gehasht und signiert werden. Schließlich werden die Digests der Elemente nicht in die Signatur eingebettet und übertragen, da diese durch den Überprüfer der Signatur ohnehin direkt aus den Elementen errechnet werden müssen.

Vergleicht man die Signaturen aus den Beispielen in den Listings A.3 und A.4 bzw. aus Abbildung 3.1, ergeben sich die Unterschiede wie in Tabelle 3.1.

	XML Signature	WSDD Compact Signature
Anzahl Elemente	41	1
Anzahl Elementattribute	24	5
Größe in Byte	**2479**	**246**

Tabelle 3.1: Vergleich von XML Signature und WSDD Compact Signature

Bei identischem Informationsgehalt ist die kompakte Darstellung um den Faktor zehn kleiner und enthält nur ein Element mit fünf Attributen, wodurch auch die Kosten für den XML-Parser gesenkt werden. Während bei einer XML-Signature für jedes zusätzlich zu signierende Element weitere ca. 350 Byte in 9 Elementen mit 8 Attributen hinzukommen, bleibt die Größe der WSDD Compact Signature nahezu konstant, da ausschließlich die IDs der zu signierenden Elemente hinzukommen, also ca. 5-10 Byte.

Die beschriebene Art und Weise der Bildung einer WSDD Compact Signature lässt es zu, dass „klassische“ und kompakte Signaturen ineinander überführt werden können. Somit ist die Basis der Kompatibilität zu bestehenden WS Security-Implementierungen geschaffen.

3.2.2 Kompaktes Einbetten von Chiffren

Mit dem kompakten Format zum Anhängen von Signaturen an eine SOAP- bzw. XML-Nachricht existiert bereits einer der beiden benötigten Teile für ein anvisiertes WS Compact Security. Für den zweiten benötigten Teil – die Einbettung von Chiffren – existierte noch keine kompakte Lösung. Dieses wurde im Rahmen dieser Arbeit in Anlehnung an die WSDD Compact Signatures

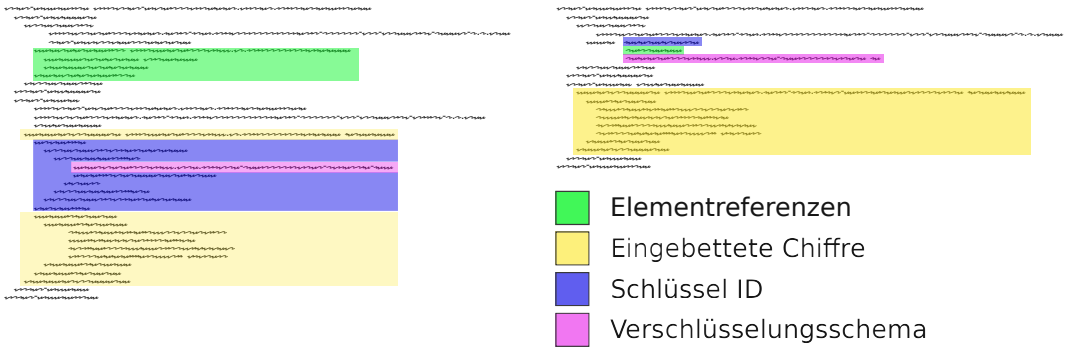


Abbildung 3.2: Vergleich von XML-Encryption (links) und kompakter Darstellung (rechts)

entwickelt und wird nun auf die gleiche Weise mit seiner „klassischen“ Vorlage – hier XML-Encryption – verglichen. Eine farblich markierte Zuordnung in den schematischen Darstellungen zeigt Abbildung 3.2. Die zugrundeliegenden Nachrichten können detailliert im Anhang in den Listings A.5 und A.6 betrachtet werden.

Im Gegensatz zum Anhängen der Signatur, die durch ein Element im `<Security>`-Header-Element dargestellt wird, beinhaltet das für die Verschlüsselung verantwortliche Header-Element `<ReferencesList>` lediglich die Referenzen auf die Elemente der originalen Nachricht, deren Inhalt verschlüsselt wurde (grün). Im Body der Nachricht befindet sich dann anstelle des Klartextes ein `<EncryptedData>`-Element. Es enthält neben der eigentlichen Chiffre (gelb) Angaben über das verwendete Verschlüsselungsschema (Algorithmus und z.B. Schlüssellänge, magenta) und den verwendeten Schlüssel (violett).

Beim Einbetten von Chiffren ist das Maß der Redundanz nicht so hoch wie beim Anhängen von Signaturen, wodurch auch die prozentuale Verringerung der Größe weniger deutlich ausfällt. Dennoch lässt sich die Größe durch Verwendung einer kompakteren Schreibweise und durch die Festlegung, dass alle genannten Elemente mit dem gleichen Schema und dem selben Schlüssel verschlüsselt werden, reduzieren. Somit können alle Angaben in einem neuen Header-Feld `<Enc>` zusammengefasst werden, im Body sind nur noch die eigentlichen Chiffren zu finden. Einen direkter Vergleich der Schemata in den Listings A.5 und A.6 bzw. in Abbildung 3.2 ergibt die Gegenüberstellung in Tabelle 3.2.

	XML Encryption	Compact Encryption
Anzahl Elemente	8	3
Anzahl Elementattribute	4	4
Größe in Byte	**539**	**227**

Tabelle 3.2: Vergleich von XML Encryption und Compact Encryption

Auch hier ist eine Größenreduzierung um mehr als die Hälfte zu erkennen und die Anzahl der durch einen XML-Parser zu verarbeitenden Elemente ist ebenfalls halbiert worden. Die durch das WSDD Compact Signature Format begünstigte Kompatibilität zu existierenden WS Security Implementierungen wird durch das Compact Encryption Schema nicht beeinträchtigt, da es sich ebenfalls in das durch XML-Encryption definierte Schema überführen lässt.

3.2.3 Kombination zu WS Compact Security und Vergleich mit TLS

Mit dem neu entwickelten Compact Encryption Schema sind alle Bauteile für ein WS Compact Security vorhanden. Die Listings 3.1 und 3.2 zeigen eine einfache SOAP-Nachricht und wie durch WS Compact Security ihre Vertraulichkeit, Integrität und Authentizität gewährleistet werden.

```
1 <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
2   <SOAP-ENV:Body>
3     <ns:single-string-echo xmlns:ns="ws4d:ptest">
4       <ns:in>01234</ns:in>
5     </ns:single-string-echo>
6   </SOAP-ENV:Body>
7 </SOAP-ENV:Envelope>
```

Listing 3.1: Beispiel einer einfach SOAP Test-Nachricht

```
1 <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
2   <SOAP-ENV:Header>
3     <d:Security xmlns:d="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01
          ">
4       <d:Enc KeyId="ZHVtbXk="
5              Scheme="http://www.ws4d.org/ws4d-sec/2011/08/aes">
6       </d:Enc>
7       <d:Sig KeyId="ZHVtbXlkdW1teWR1bW15ZHVtbXk="
8              Refs=""
9              Scheme="http://www.ws4d.org/ws4d-sec/2011/08/aes-cbc-sha-hmac"
```

```
10              Sig="dtyuCvR713aKt3I4Fv/4bsUQcT7u6JkNvVL0yZFZwE0=">
11        </d:Sig>
12      </d:Security>
13    </SOAP-ENV:Header>
14    <SOAP-ENV:Body>
15      <d:EncryptedData xmlns:d="http://docs.oasis-open.org/ws-dd/ns/discovery
            /2009/01" Id="bodyID">
16        <d:CipherData>ZS40Yxz0H72gN6l00lNbtKNXw/E0Zl Y0YpS2JpEK2q1fkgCPy0QKil+
              sTSlZcnyWbbBPFotAEsKKkjlZkkRBD/AqP+q0aQ38RRhrAFg+
              mTtV1CXQqIMiNSkAzdJ9ExsJ</d:CipherData>
17      </d:EncryptedData>
18    </SOAP-ENV:Body>
19  </SOAP-ENV:Envelope>
```

Listing 3.2: Die Nachricht aus Listing 3.1 gesichert mit WS Compact Security

In Listing 3.2 werden zwei Konventionen ersichtlich, die Teil des in dieser Arbeit spezifizierten WS Compact Security Schema sind. Bleiben die Elementreferenzattribute `Sig/@Refs` oder `Enc/@Refs` leer oder fehlen sie ganz, so bedeutet das implizit, dass die gesamte Nachricht, also nur das Element `<Envelope>` in die Signaturbildung eingeht bzw. dass ausschließlich das `<Body>`-Element verschlüsselt wird.

Das Signieren des gesamten `<Envelope>`-Elements ist nur möglich, wenn kein Szenario unterstützt werden muss, in dem verschiedene Absender der Nachricht signierte Teile hinzufügen müssen. In diesem Fall entsteht kein Aufwand für das Parsen und Zerlegen der Nachricht, um die Digests der richtigen Nachrichtenteile berechnen zu können.

Den Body einer Nachricht zu verschlüsseln ist außerdem der gängigste Fall, da dieser als einziges Element sensible Informationen beinhalten sollte. Auch hier ist das Auslassen der Referenzen nur möglich, wenn nicht mehrere Absender einer Nachricht verschiedene verschlüsselte Abschnitte hinzufügen müssen.

Trifft eine der Bedingungen nicht zu, so können wie gewohnt die Elementreferenzen angegeben werden. Die Kompatibilität zu „klassischen“ WS Security Implementierungen bleibt ebenfalls erhalten.

Setup für Performance Messungen

Durch die Reduzierung der Nachrichtengrößen wird der Nachteil von WS Security gegenüber TLS reduziert. Es bleibt die Fragestellung nach dem Vergleich der Performance. Zu diesem Zweck wurde das WS Compact Security Sche-

ma als Plug-in für das offene C-Web-Services-Framework gSOAP ([van14]) in Version 2.8.3 umgesetzt. Ebenfalls in gSOAP wurde zu Testzwecken ein sogenannter Echo-Web-Service implementiert, dessen Aufgabe es ist, einen oder mehrere Strings als Parameter von einem Client anzunehmen und unverändert als Antwort zurück zu geben. Der zugehörige Client wurde ebenfalls in gSOAP implementiert. Seine Aufgabe ist es, eine definierte Anzahl Requests an den Service zu richten, auf die Antwort zu warten und die benötigte Zeit zu messen. Ein Request besteht je nach Testfall aus einer der Nachrichten in den Listings A.7, A.9 oder A.11 und die Antworten durch den Echo-Service sind entsprechend in den Listings A.8, A.10 und A.12 dargestellt. Sowohl die Requests als auch die Responses unterscheiden sich durch die Größe der enthaltenen Nutzlast (134-353 Byte). Client- und Server- Software sind in der Lage, die Testnachrichten unverschlüsselt, über einen TLS-Kanal, mittels WS Compact Security oder mittels WS Security gesichert zu empfangen und zu versenden. Wie eingangs erwähnt, kam als Implementierung für WS Compact Security ein Plug-in zum Einsatz, das im Rahmen dieser Arbeit entwickelt wurde. Die Implementierung von WS Security, gegen die WS Compact Security antritt, existierte bereits in Form eines Plug-ins als Bestandteil von gSOAP und setzt die Spezifikation WS-Security vollständig um.

Die Client-Software wurde auf einem PC mit einer Intel Core i7 870 CPU mit 1,2GHz Taktfrequenz und 8GB Arbeitsspeicher ausgeführt. Als Betriebssystem kam KUbuntu Linux 11.04 zum Einsatz. Die Quelldateien wurden mit der Compiler-Suite GCC in Version 4.5.2 übersetzt.

Die Server-Software, die den Echo-Service anbietet, wurde auf einer Embedded-Linux-Plattform ausgeführt. Hierzu kam ein FoxBoard LS832 von ACME Systems zum Einsatz. Es beherbergt eine Axis ETRAX 100LX RISC CPU, 32MB Arbeitsspeicher und 8MB Flash-Speicher, der das Betriebssystem und Anwendungen enthält. Die Übersetzung der Quellen übernahm der CRIS GCC Cross-Compiler in Version 3.2.1 (Release 1.64) des Herstellers Axis.

Beide Compiler wurden angewiesen, die Kompilate hinsichtlich ihrer Code-Größen zu optimieren (Flag -Os). Die Kommunikation zwischen Client und Server, also zwischen PC und Foxboard, fand über eine dedizierte Ethernetverbindung statt, um äußere Störeinflüsse durch bestehende Netzwerklast aus-

	Lines of Code
WS Compact Security Plugin	2147
Test-Programme (Client und Server)	911
WS Security Plugin	4667

Tabelle 3.3: Lines of Code der einzelnen Anwendungsteile, WS Security Implementierung zum Vergleich

	No Security	TLS	WS Security	WS Compact Security
PC (amd64)	175	193	373	205
AXIS RISC CPU	197	1158	1273	837

Tabelle 3.4: Codegrößen der Prototyp-Implementierungen in kByte, WS Security Implementierung zum Vergleich

zuschließen. Die Verbindungsgeschwindigkeit wurde durch die Fähigkeiten des Foxboards auf 100MBit/s begrenzt.

Vor der Aufnahme von Messwerten wurden die entwickelten Test-Anwendungen und das entwickelte Plug-in sorgfältig mit den Werkzeugen valgrind bzw. memgrind untersucht, um Speicherlecks als mögliche Ursache für Performanceverluste aufzuspüren und zu eliminieren.

Die ebenfalls quelloffene Bibliothek OpenSSL in Version 1.0.1 stellte sowohl die TLS-Funktionalität als auch die benötigten kryptografischen Routinen bereit. Als Verschlüsselungsalgorithmen wurden die Strom-Chiffre RC4 und die Block-Chiffre AES-CBC mit einer Schlüssellänge von je 128 Bit untersucht. Dies sind die jeweils relevantesten Vertreter der jeweiligen Kategorien und werden beispielsweise von der DPWS-Spezifikation erfordert bzw. empfohlen ([OAS09a, cpt.6]). Als Hash-Funktion für die Berechnung von MACs und Signaturen kam SHA-1 zum Einsatz.

Die entstandenen Lines of Code sowie die Größen der prototypischen Implementierungen sind in den Tabellen 3.3 und 3.4 dargestellt.

Messmethode und Ergebnisse

Um das entwickelte WS Compact Security Schema mit TLS als Stand der Technik zu vergleichen, wurde als Metrik die sogenannte *Round-Trip-Time*

		SOAP *(+HTTP)*	TLS		WS CSec	
			RC4	AES	RC4	AES
Testfall 1	Request	218 *(382)*	238	245	749	777
	Response	236 *(372)*	222	229	773	801
Testfall 2	Request	263 *(427)*	248	261	809	841
	Response	281 *(417)*	232	245	832	865
Testfall 3	Request	437 *(601)*	293	309	1041	1077
	Response	455 *(591)*	277	293	1065	1097

Tabelle 3.5: Größen der Testnachrichten in Byte unverschlüsselt, unverschlüsselte TCP-Payload, verschlüsselte TCP-Payload und als SOAP-Message mit WS Compact Security

(RTT) verwendet. Die RTT bezeichnet hierbei die Zeit, die vom Stellen einer Anfrage durch den Client bis zum Eintreffen der Antwort vom Server vergeht.

Um den Einfluss von Verzögerungseffekten z.B. durch Hintergrundprozesse zu minimieren, stellt der Client 10.000 identische Anfragen an den Server. Anstatt die gesamte Zeitspanne vom ersten bis zum letzten Request zu ermitteln und das arithmetische Mittel zu errechnen, wurde jede RTT einzeln gemessen, um anschließend den Median der Werte zu berechnen, da dieser unanfälliger für statistische Ausreißer ist.

Tabelle 3.5 zeigt zunächst die Größen der einzelnen Testnachrichten und den jeweiligen Antworten. Angegeben sind dabei die Größen des SOAP-Envelopes und dahinter in Klammern die der jeweiligen TCP-Payload. Nachfolgend stehen die Größen der TCP-Payloads (also der TLS-Records) für den Fall dass die Nachrichten über einen TLS-Kanal versendet werden. Die Verwendung von AES führt stets zu größeren Nachrichten. Die liegt darin begründet, dass Blockchiffren wie AES den Klartext auf ein ganzzahliges Vielfaches der Blockgröße padden müssen. Dies geschieht jeweils für den Klartext sowie für den Digest, der zur Signatur verschlüsselt wird. Darüber hinaus müssen die entstehenden binären Daten base64-codiert werden, um in eine SOAP-Nachricht eingebettet zu werden, was die Größe weiter erhöht. Es fällt auf, dass die TLS Records kleiner sind als die unverschlüsselt übertragenen Nachrichten. Die Größe verringert sich demnach um 40% bis 50% wobei der Effekt mit steigender Klar-

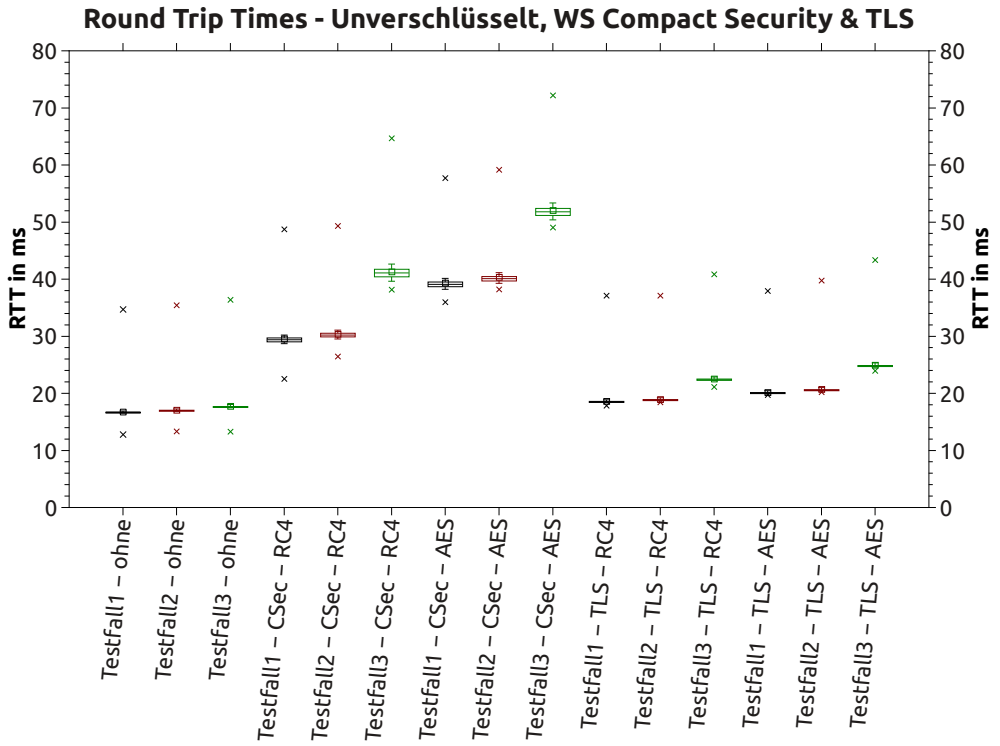


Abbildung 3.3: RTT für unverschlüsselte Nachricht und Sicherung mit WS Compact Security und TLS

textgröße deutlicher wird. Der Grund für dieses Phänomen liegt darin, dass die Daten in TLS Records vor ihrer Verschlüsselung komprimiert werden. Die Anwendbarkeit dieses Vorgehens auf WS Compact Security wird im folgenden Abschnitt untersucht. Schließlich zeigt sich, dass die Nachrichtengröße durch die Verwendung von WS Compact Security um den Faktor 2,5 bis 3,5 steigt, wobei der Effekt mit steigender Klartextgröße geringer ist, was an der beinahe konstanten Größe des Security-Headers liegt.

Die ermittelten Nachrichtengrößen spiegeln sich in den Laufzeiten wider. Abbildung 3.3 zeigt den Vergleich der RTTs ganz ohne Verschlüsselung und für die Nutzung von WS Compact Security bzw. TLS. Die jeweiligen Mediane der RTT sind darüber hinaus in Tabelle 3.6 zum direkten Vergleich gegenübergestellt.

Es ist erkennbar, dass die Nutzung von TLS gegenüber dem vollständigen Verzicht auf Verschlüsselung eine Erhöhung der RTT um den Faktor 1,1 bis 1,3 zur Folge hat, trotz der stark verringerten Nachrichtengrößen. Die Nutzung

		Round Trip Times in ms		
		Testfall 1	Testfall 2	Testfall 3
ohne		16.7	17.0	17.6
WS CSec	RC4	29.4	30.2	41.1
	AES	39.1	40.1	51.8
TLS	RC4	18.5	18.8	22.4
	AES	20.0	20.6	24.8

Tabelle 3.6: RTT (Mediane) bei Nutzung von WS Compact Security verglichen mit TLS

des Compact Security Schemas erhöht die RTT um den Faktor zwei bis drei, je nach Größe der Nachricht und verwendetem Verschlüsselungsalgorithmus.

Der Grund hierfür liegt in einem grundlegenden Unterschied des Record-Protokolls von TLS und den bisher gezeigten Ansätzen auf Anwendungsebene. Die Vorgänge zum Verschlüsseln einer Nachricht und zur Berechnung ihrer Signatur sind bei WS Security bzw. WS Compact Security vollständig unabhängig. Das hat den Vorteil, dass es möglich ist, Nachrichten zwar signiert jedoch unverschlüsselt zu versenden, wenn Integrität und Authentizität einer Nachricht zwar gewährleistet sein müssen, Vertraulichkeit jedoch keinen Sicherheitsgewinn brächte.

TLS verfährt hier anders. Anstatt Verschlüsselung und Signaturbildung getrennt voneinander zu behandeln, wird vor der Verschlüsselung ein Message Authentication Code (MAC) an die Nachricht angehängt. Ein solcher MAC ist gleichbedeutend mit einer Hash-Wert-Berechnung und entsprechend ressourcenschonend. Somit muss zur Erzeugung eines Security Records nur eine Chiffre berechnet werden, bei der Verwendung von WS Compact Security hingegen sind es zwei.

3.3 Performance-Steigerung durch WS Security Records

Inspiriert vom Record-Protokoll von TLS wurde im Zuge dieser Arbeit daher ein weiteres Nachrichtenformat zum sicheren Datentransfer auf Anwendungs-

Type		WS CSec		WS SecRec		TLS	
		RC4	AES	RC4	AES	RC4	AES
Testfall 1	Request	749	777	508	524	238	245
	Response	773	801	532	544	222	229
Testfall 2	Request	809	841	568	588	248	261
	Response	832	865	592	598	232	245
Testfall 3	Request	1041	1077	800	820	293	309
	Response	1065	1097	825	844	277	293

Tabelle 3.7: Vergleich der Nachrichtengrößen von WS Compact Security, WS Security Records und TLS

ebene entwickelt, die sogenannten Security Records. Sie ersetzen den Body einer Nachricht durch ein `<Record>`-Element wie in Listing B.8 dargestellt.

Die Attribute des `<Record>`-Elements sind inhaltlich die selben wie die Attribute des `<Enc>`- und des `<Sig>`-Elements des WS Compact Security Schemas allerdings konsolidiert in einem Element. Der Unterschied besteht in dem `CipherData`-Attribut. Dieses enthält die Chiffre einer Struktur, die sowohl den Inhalt des `<Body>`-Elements, als auch den Digest aller in `SigRefs` referenzierten Elemente enthält.

Auf diese Weise ermöglicht das WS Security Record Format eine noch kompaktere Darstellung der zu sichernden Nachrichtenteile. Darüber hinaus ist somit nur ein Verschlüsselungsdurchlauf nötig, um neben der Vertraulichkeit auch Nachrichtenintegrität und -authentizität zu gewährleisten.

3.3.1 Vergleich der WS Security Records mit TLS

Um einen Vergleich zwischen den hier spezifizierten WS Security Records und TLS vornehmen zu können, werden zunächst die entstehenden Nachrichtengrößen verglichen.

Zunächst zeigt sich, dass das kompaktere Format erwartungsgemäß einen Vorteil gegenüber WS Compact Security darstellt (konstante Einsparung von ca. 250 Byte). Die WS Security Records sind jedoch noch immer merklich größer als bei einem Versand der ursprünglichen Nachrichten über TLS. Die Ursachen liegen hier einerseits in der Kompression der Daten, die TLS vor-

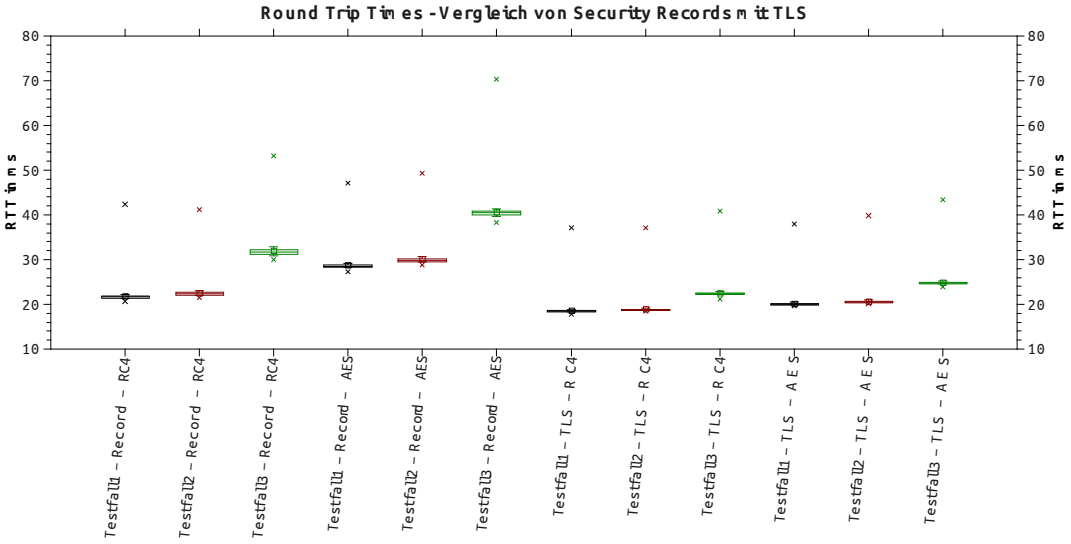


Abbildung 3.4: RTT für WS Compact Security, WS Security Records und TLS

		Round Trip Times in ms		
		Testfall 1	Testfall 2	Testfall 3
WS Compact Security	RC4	29.4	30.2	41.1
	AES	39.1	40.1	51.8
WS Security Records	RC4	21.6	22.4	31.8
	AES	28.5	29.9	40.5
TLS	RC4	18.5	18.8	22.4
	AES	20.0	20.6	24.8

Tabelle 3.8: RTT (Mediane) von WS Security Records verglichen mit WS Compact Security und TLS

nimmt, wie auch in der Tatsache, dass SOAP ein ASCII-basiertes Protokoll ist, weshalb binäre Daten wie z.B. eine Chiffre base64-codiert werden müssen was ihre Größe um ca. 33% erhöht ([Eng02]).

Die Laufzeiten sind in Abbildung 3.4 zu sehen und TLS gegenübergestellt. Die Mediane sind in Tabelle 3.8 zusammengefasst. Auch hier erfolgt eine Gegenüberstellung mit TLS sowie mit den durch WS Compact Security erzielten Werten.

Im Vergleich zu WS Compact Security ergeben sich bei den WS Security Records Reduktionen der RTTs um 22%–27%, wobei die Ersparnis mit steigender Nutzlastlänge abnimmt. Verglichen mit TLS liegen die Paketlaufzeiten

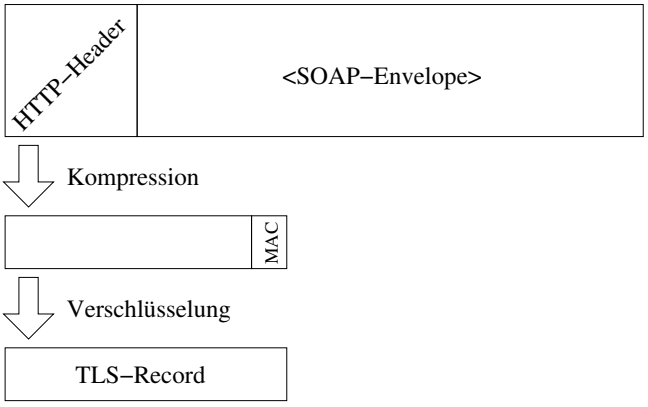


Abbildung 3.5: Einordnung der Kompression im Ablauf der Erstellung eines Records in TLS

z.T. unmerklich darüber. Für kleine Nutzlasten liegt der Faktor bei 1,2 – 1,4, bei größeren Nutzlasten bei 1,4 – 1,6. Der verringerte Verschlüsselungsaufwand kann demnach die größeren Nachrichten nicht vollständig kompensieren.

3.3.2 Kompression der Daten

Mehrfach angesprochen wurde der Performance-Vorteil, den TLS erreicht, indem es die Daten vor dem Versand komprimiert [Eck08]. Das vollständige Vorgehen ist in Abbildung 3.5 zusammengefasst. Die Kompression findet demnach noch vor der Berechnung des MAC und somit vor der Verschlüsselung statt. Grund dafür ist, dass eine Chiffre idealerweise eine enorm hohe Entropie aufweist. Nachrichten mit hoher Entropie lassen sich jedoch nicht oder kaum komprimieren.

Zur Kompression seiner Daten kommt in TLS der Deflate-Algorithmus zum Einsatz ([Hol04], [Deu96]), daher wurde die Implementierung des WS Compact Security Plug-ins und der WS Security Records um die Möglichkeit der Kompression erweitert. Bei aktivierter Kompression wurde dabei der gesamte HTTP-Layer inklusive enthaltener SOAP-Nachricht vor Versand komprimiert und bei Empfang unmittelbar dekomprimiert, um es den verarbeitenden Software-Teilen zu übergeben.

Die Fähigkeit, Daten mit dem Deflate-Algorithmus zu komprimieren und zu dekomprimieren übernimmt im entwickelten Prototypen die Open-Source-

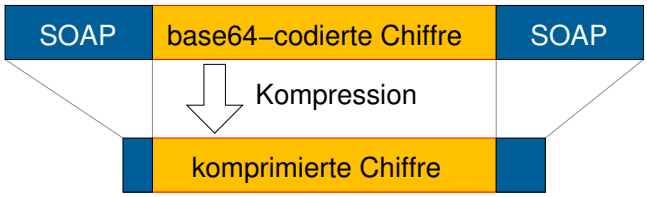


Abbildung 3.6: WS Security Records: Erst verschlüsseln, anschließend komprimieren

Bibliothek zlib ([GA14]) in Version 1.2.3.4[1]. Die Bibliothek akzeptiert zwei Parameter, mit denen die Kompression konfiguriert werden kann. Die *Chunk-Size* spezifiziert die Größe des Zwischenspeichers, der bei der Kompression genutzt wird. Seine Größe kann zwischen 1024, 2048, 4096 und 8192 gewählt werden. Größeres Chunks versprechen eine schnellere Kompression. Der zweite Parameter ist das nicht näher spezifizierte Kompressionslevel zwischen 0 und 9, wobei kleine Werte hinsichtlich Geschwindigkeit und große Werte hinsichtlich Kompressionsgrad optimieren. Level 0 bedeutet keine und Level 9 die stärkste Kompression.

Für jede Kombination aus Kompressionsstufe und Chunk-Größe wurden Laufzeiten und Nachrichtengrößen für alle drei Testnachrichten (Listings A.7, A.9 und A.11) und jeweils für RC4 und AES aufgenommen. Die ermittelten Nachrichtengrößen und Laufzeiten finden sich im Anhang in den Tabellen A.1 für RC4 und A.2 für AES. Es zeigt sich, dass die Nachrichtengrößen – fast unabhängig vom Kompressionsgrad – um ca. 30% reduziert werden. Auch die Länge der verschlüsselten Payload spielt nur eine untergeordnete Rolle, man erkennt jedoch, dass kurze Payloads zu einer etwas höheren Kompressionsrate führen als lange Payloads. In den untersuchten Beispielen liegt der Unterschied bei ca. 3%.

Für die geringe Kompressionsrate ist der Umstand verantwortlich, dass bei Verschlüsselung der HTTP-Ebene die Entropie der Nachricht durch die enthaltene Chiffre bereits sehr hoch ist und nur die nicht chiffrierten Teile des SOAP-Envelopes komprimiert werden können (vgl. Abbildung 3.6). Hier wird demnach zuerst verschlüsselt und anschließend komprimiert. Der umgekehrte Weg wäre, die Nutzlast erst zu komprimieren und anschließend zu verschlüs-

[1] Anm.: Es handelt sich dabei tatsächlich um die zu der Zeit aktuelle Versionsnummer der Bibliothek.

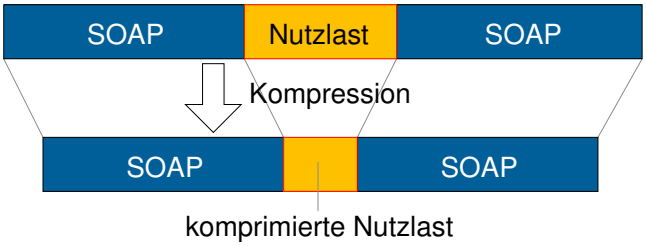


Abbildung 3.7: WS Security Records: Erst komprimieren, anschließend verschlüsseln

seln, ganz so, wie es bei TLS gehandhabt wird. Dann wird jedoch nur ein kleiner Teil des ursprünglichen SOAP-Envelopes komprimiert, und ein erheblicher Teil wird unkomprimiert versendet (vgl. Abbildung 3.7).

Betrachtet man schließlich die Round Trip Times, so wird deutlich, dass die Kompression der Daten keinen Performance-Vorteil erbringt. Werden die Daten mit RC4 verschlüsselt, erhöhen sich die RTT bei aktivierter Kompression um 21%–24,5%, bei einer Verschlüsselung mittels AES erhöhen sie sich um 16,3%–17,5%. Der Vorteil durch geringere Nachrichtengrößen wird demnach vom Mehraufwand durch die Kompression und Dekompression übertrumpft.

3.4 Zusammenfassung und Diskussion

Stand der Technik für den Aufbau verschlüsselter Kommunikationskanäle durch eingebettete vernetzte Systeme ist die Protokollgruppe Transport Layer Security. In diesem Kapitel wird gezeigt, dass der Grund dafür in seiner hohen Verbreitung auf PC- und Server-Systemen liegt, dass TLS jedoch für verteilte eingebettete Systeme nicht die ideale Wahl ist. Das liegt an seiner Abhängigkeit von X.509-Zertifikaten zur Authentifizierung von Kommunikationspartnern, den damit einhergehenden Verteilungsproblemen, der fehlenden Ende-zu-Ende-Sicherheit bei Multihop-Szenarien und der Abhängigkeit von einem Transportprotokoll.

Als Lösungsansatz wird daher zunächst ein Mechanismus auf Anwendungsebene auf Basis von WS Security zur Gewährleistung von Integrität, Authentizität und Vertraulichkeit vorgestellt. Der WS Compact Security genannte Mechanismus unterscheidet sich von seiner Vorlage in einer deutlich kompakteren Repräsentation der zusätzlichen Nachrichtenteile. Im Fall des Beispiels

aus Listing A.7 erhöht WS Compact Security die Nachrichtengröße nur auf 749 statt 1808 Byte. Das entspricht einer Reduktion von beinahe 60%.

Vergleicht man WS Compact Security mit TLS als Stand der Technik, so wird deutlich, dass die Nachrichten bei dem hier vorgestellten Ansatz 2,5 – 3,5 mal größer sind als bei Verwendung von TLS. Auch die Round Trip Times erhöhen sich um die Faktor 2 – 3 gegenüber der Verwendung von TLS. Die Gründe liegen in der getrennten Behandlung von Nachrichtenintegrität / -authentizität und Vertraulichkeit sowie in der fehlenden Kompression.

Die Verdopplung bis Verdreifachung der Reaktionszeit kann jedoch einen akzeptablen Kompromiss darstellen, um von den funktionalen Vorteilen, die WS Compact Security gegenüber TLS bietet, zu profitieren. So erlaubt WS Compact Security alternative Authentifizierungsmechanismen, wodurch die Notwendigkeit des Austauschs und der Haltung von X.509-Zertifikaten entfällt. Stattdessen können Methoden wie eine direkte Authentifizierung durch Austausch einer Einweg-PIN genutzt werden. Die Verschlüsselung der Nutzlast auf Anwendungsebene begünstigt zudem eine sichere Datenhaltung und ermöglicht Ende-zu-Ende-Verschlüsselung in Multihop-Szenarien. Schließlich entfällt die Abhängigkeit von einem Transportprotokoll. Ein Konzept auf Anwendungsebene kann demnach ebenso mit Technologien eingesetzt werden, die die ISO-OSI-Layer-Architektur nicht berücksichtigen, so wie z.B. Bluetooth und ZigBee.

Das WS Compact Security Schema kann ohne Neuberechnung von Nachrichtenteilen in seine klassische Form überführt werden, was ein bedeutender Punkt ist, wenn die Kompatibilität zu bestehenden WS Security Implementierungen von Belang ist. Kann auf diese Kompatibilität verzichtet werden, und will man ausschließlich den Body einer Nachricht verschlüsseln, so können alternativ die WS Security Records eingesetzt werden. Sie betrachten Vertraulichkeit und Nachrichtenintegrität / -authentizität geschlossen und bedürfen daher nur eines Verschlüsselungsdurchlaufs. Außerdem lassen sie eine noch kompaktere Repräsentation zu. Gegenüber TLS liegt ihr Geschwindigkeitsnachteil nur noch bei Faktor 1,2 – 1,4 für kleine Nutzlasten bzw. 1,4 – 1,6 bei größeren, sie bieten jedoch die selben funktionalen Vorteile.

Es konnte gezeigt werden, dass Kompression in dem gezeigten Messaufbau zwar die Nachrichtengröße reduziert, die Round Trip Times jedoch erhöht.

Der Grund liegt dabei in der geringen Kompressionsrate aufgrund der hohen Nachrichtenentropie zum Zeitpunkt der Kompression. Es sei darauf hingewiesen, dass die Messungen über eine dedizierte Ethernet-Verbindung durchgeführt wurden. Es ist denkbar, dass bei drahtlosen Verbindungen mit geringer Übertragungsrate Kompression ein probates Mittel darstellt, um die Funkübertragungszeiten und somit den Leistungsbedarf zu minimieren. Dieser Zusammenhang konnte im Rahmen dieser Arbeit nicht mehr untersucht werden.

Kapitel 4

Erhöhung der Nutzerfreundlichkeit bei direkter Authentifizierung

Der in diesem Kapitel beschriebene Vorschlag zur Überbrückung inkompatibler Nutzerschnittstellen, die zugrundeliegenden Protokolle sowie der entstandene Prototyp wurden im Rahmen der Anfertigung dieser Dissertation bereits veröffentlicht:

> *Sebastian Unger, Dirk Timmermann: „Bridging the UI Gap for Authentication in Smart Environments“ in 2014 IEEE Symposium on Computers and Communication (ISCC), 2014. DOI: 10.1109/ISCC.2014.6912579 [UT14]*

In diesem Kapitel wird ein Konzept vorgestellt, das eventuelle Inkompatibilitäten von Ein- und Ausgabemöglichkeiten bei einer direkten Authentifizierung von zwei Geräten überbrückt. Das Konzept soll Kosten beim Bau von Komponenten in intelligenten Umgebungen verringern und gleichzeitig die Benutzbarkeit direkter Authentifizierungsvorgänge erhöhen.

Im Anschluss an die Motivation werden der grundlegende Ansatz und das zugrundeliegende kryptografische Protokoll erläutert. Die Beschreibung des Prototypen enthält die einzelnen Komponenten sowie die entstandenen Kommunikationsprotokolle. Nach einer Beschreibung der Abläufe für den Nutzer werden Erweiterungen diskutiert.

4.1 Motivation und Lösungsansatz

Wie in Kapitel 2.2.2 beschrieben, handelt es sich bei dem Vorgang der Authentifizierung um eine gegenseitige Identifizierung mit anschließendem oder gleichzeitigem Austausch oder Aufbau von Schlüsselmaterial und man unterscheidet die vermittelte bzw. delegierte und direkte Authentifizierung (vgl. Abbildung 2.6, S. 35). Während bei der vermittelten Authentifizierung (z.B. bei PKI, Mechanismen aus WS Trust oder beim Kerberos-Protokoll, [NYHR05]) alle Vertrauensbeziehungen über definierte vertrauenswürdige Instanzen delegiert werden (vgl. Abb. 2.6 b)), werden bei der direkten Authentifizierungen alle Beziehungen direkt zwischen den jeweiligen Partnern hergestellt.

Die vermittelte Authentifizierung hat den Vorteil, dass alle Teilnehmer nur eine Vertrauensbeziehung zu einer vertrauenswürdigen Partei unterhalten müssen, was zu einem geringen Speicherbedarf führt. Hierdurch entsteht jedoch der Nachteil, dass eine Infrastruktur aufgebaut werden muss. Hierfür benötigt jeder Teilnehmer seine eigenen Credentials sowie die der vertrauenswürdigen dritten Partei bereits zum Zeitpunkt ihrer Ausbringung ins Feld. Zudem steigt der Kommunikationsaufwand, da für jede Anfrage eines Clients A an einen Service B bzw. für jede Sitzung der Client ein Token von der vertrauenswürdigen Partei T anfordern muss, das seine Authentizität gegenüber B bestätigt. Schließlich beinhaltet eine solche Lösung immer einen Single-Point-of-Failure, der die Ausfallsicherheit empfindlich beeinträchtigt. Fällt die vertrauenswürdige Partei aus, ist keine gesicherte Kommunikation mehr möglich. Existierende Middlewares, die auf dieser Form der Authentifizierung basieren, sind beispielsweise Virtus ([CBB+12]), Peces ([HHA+10]) und SM4ALL ([BDCM+09]).

Demgegenüber steht die direkte Authentifizierung. Sie basiert immer auf dem Out-of-Band-Austausch (OOB) einer Information. OOB bedeutet, dass diese Information über einen gesonderten Kanal übertragen wird. Ein Beispiel hierfür ist z.B. die Installation von vertrauenswürdigen Zertifikaten in den Speicher eines Geräts durch den Hersteller. Das ist zwar grundsätzlich auch zur Laufzeit im Feld durch den Nutzer möglich, jedoch ist es komfortabler, eine kurze Einweg-PIN OOB zu übertragen und dann ein kryptografisches Protokoll wie in Kapitel 2.2.2 auszuführen, wie z.B. ein Challenge-Response-Verfahren. Dieser OOB-Austausch kann im einfachsten Fall eine numerische

PIN sein, die von einem Gerät dargestellt wird und über die Tastatur des anderen Geräts eingegeben wird. Eine solche PIN kann jedoch über verschiedenste physikalische Kanäle übertragen werden. Umfassende Überblicke über unterschiedliche Methoden sind in [KSTU09] und [KST+09] zu finden. Möglich sind z.B. sichtbares Licht ([SEKA06]), (2D)-Barcodes ([M+05]) oder Audio-Kanäle ([G+06, STU08]). Neben serieller Infrarot- oder Nahfeldkommunikation ist auch die Verwendung von Beschleunigungssensoren denkbar ([MG07]). Ein Nachteil direkter Authentifizierung liegt in der Tatsache, dass jede Herstellung einer Vertrauensbeziehung manuell durch den Nutzer erledigt werden muss. Außerdem müssen die einmal aufgebauten Beziehungen durch die Geräte selbst verwaltet werden, was evtl. einen hohen Speicherbedarf zur Folge hat.

Die Vorteile von direkter und vermittelter Authentifizierung können in hybriden Lösungen kombiniert werden. Existiert die Möglichkeit der vermittelten Authentifizierung, so wird sie wahrgenommen und läuft demnach transparent und unbemerkt für den Nutzer ab. Sie resultiert dann in einer direkten Vertrauensbeziehung zwischen den beteiligten Geräten. Wird sie lange nicht genutzt, kann sie zugunsten benötigten Speicherplatzes entfernt und bei Bedarf wiederhergestellt werden. Ist keine vermittelte Authentifizierung möglich, so kann stattdessen eine direkte Authentifizierung mittels eines OOB-Austauschs stattfinden.

Es bleibt jedoch ein Problem beim OOB-Austausch einer Einweg-Pin zum Zweck der direkten Authentifizierung bestehen. So benötigen zwei Geräte zum Austausch kompatible Schnittstellen, wie z.B. ein numerisches Display und Zifferntastatur, Lautsprecher und Mikrofon oder Infrarotsender und -empfänger. Besteht diese Kompatibilität nicht, so ist es auch nicht möglich, die Einweg-Pin auszutauschen. Ein möglicher Ansatz zur Lösung dieses Problems könnte beinhalten, in einem Profil oder einer Konvention zu bestimmen, dass jedes Gerät beispielsweise über einen NFC-Transceiver verfügen muss, über den der OOB-Austausch stattfindet, ähnlich wie in [BCQ+10] beschrieben.

Das hätte zur Folge, dass jedes Gerät über einen solchen Transceiver verfügen muss, auch, wenn es ihn für seine primäre Funktionalität nicht benötigt. Zudem würde dieses konkrete Beispiel Mobilität von einem Großteil der Geräte erfordern, um die räumliche Nähe zwischen den Geräten herstellen zu können.

Zur Lösung dieses Problems wird in dieser Arbeit die indirekte Authentifizierung vorgestellt. Sie beruht auf der Annahme, dass in den anvisierten Szenarien ein oder mehrere Multimediageräte vorhanden sind. Bei Multimediageräten handelt es sich um z.B. Fernsehgeräte mit Settop-Boxen bzw. Smart-TVs, Internetradios, digitale Bilderrahmen, Tablet-PCs oder Smartphones. Vordergründig sind dabei ihre Eigenschaften, über eine Vielzahl an Schnittstellen zu verfügen und sich im direkten Wirkbereich ihrer Nutzer zu befinden. Im Falle der Smartphones tragen die meisten Nutzer sie üblicherweise bei sich oder haben sie immer griffbereit.

Diese Multimediageräte sollen die Übersetzung von OOB-Kanälen ineinander vornehmen, also z.B. eine numerische Pin in ein akustisches Signal umwandeln oder umgekehrt. Diese Lösung kann auf die Eigenheiten der jeweiligen Geräte eingehen und erfordert keine Peripherie in den Geräten, die für ihre eigentliche Funktion nicht erforderlich ist.

Allerdings sollen die Multimediageräte nicht einfach eine vermittelte Authentifizierung vornehmen, da sie hierfür eine vertrauenswürdige dritte Partei innerhalb des Netzwerks darstellen müssen. Stattdessen soll die Vermittlung von jedem zur Verfügung stehenden Multimediagerät vorgenommen werden können, ohne dass dies die Sicherheit beeinträchtigt. Die Vertrauensbeziehung soll zwischen den jeweiligen Endgeräten direkt entstehen. Da mobile Endgeräte wie Smartphones anfällig für Verlust oder Diebstahl sind, darf dieser Fall nicht die Sicherheit des Netzwerks gefährden, weil z.B. ausgetauschtes Schlüsselmaterial auf dem Multimediagerät zwischengespeichert wurde.

4.2 Zugrundeliegendes kryptografisches Protokoll

In Kapitel 2.2.2 wurden die werksseitige Versorgung mit Zertifikaten und das Challenge-Response-Protokoll als Möglichkeiten einer direkten Authentifizierung besprochen. Während man für ersteres alle möglichen künftigen Kommunikationspartner schon vor Auslieferung kennen muss, beinhaltet letzteres keinen Schlüsselaustausch. Im folgenden wird eine Protokollgruppe beschrieben, die von Ho in [Ho12] veröffentlicht wurde und eine gegenseitige Identifizierung samt Schlüsselaustausch ermöglicht. Sie fand Eingang in die Spezifikation

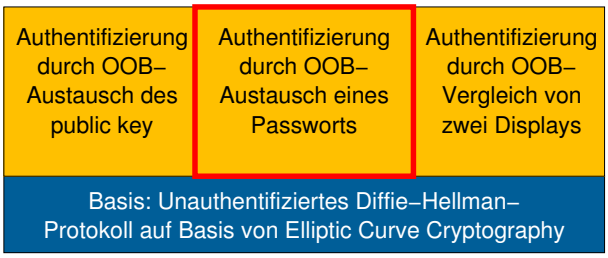


Abbildung 4.1: Prinzipielle Darstellung der von Ho in [Ho12] vorgeschlagenen Protokollgruppe

IEEE 802.15.6 für Wireless Personal Area Networks (WPANs). Anschließend werden Änderungen an dem Protokoll vorgeschlagen, um es für eine indirekte Authentifizierung nutzen zu können. Schließlich soll die Sicherheit des geänderten Protokoll neu bewertet werden.

4.2.1 Authentifizierter Diffie-Hellman-Schlüsselaufbau nach Ho

Um verschiedene Authentifizierungsmechanismen mit einer gemeinsamen Protokollbasis bereitstellen zu können, und damit Implementierungsaufwand und Code-Größe gering zu halten, hat Ho die in [Ho12] beschriebene Protokollgruppe vorgeschlagen (vgl. Abbildung 4.1). Die Grundlage dieser Gruppe bildet ein Diffie-Hellman-Schlüsselaufbau-Protokoll auf Basis elliptischer Kurven. Es unterscheidet sich lediglich durch die gewählten kryptografischen Routinen vom „klassischen“ Diffie-Hellman-Protokoll, wie es in Kapitel 2.2.2 auf Seite 21 beschrieben ist und wird im Verlauf dieses Kapitels näher beleuchtet.

Aus den öffentlich ausgetauschten Parametern sowie zwei Zufallszahlen können beide Parteien einen Sitzungsschlüssel ableiten, der niemals übertragen werden muss, und daher durch Belauschen der Kommunikation nicht herausgefunden werden kann. Ein Man-In-The-Middle-Angreifer (MITM) kann sich jedoch gegenüber beiden Parteien als die jeweils andere ausgeben und somit mit beiden einen eigenen Sitzungsschlüssel aushandeln. Dieser Angriff bleibt dabei völlig unbemerkt.

Um diesen Angriff zu unterbinden, beschreibt Ho drei Mechanismen, um den Austausch der Parameter und den damit einhergehenden Schlüsselaufbau zu authentifizieren. Im ersten Fall wird einer der öffentlichen Parameter – der

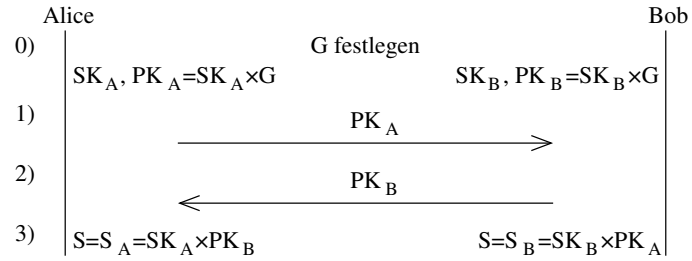


Abbildung 4.2: Ablauf eines einfachen Elliptic Curve Diffie Hellman Protokolls

Public Key eines der Teilnehmer – OOB, und somit sicher, übertragen. Die zweite Variante basiert auf dem OOB-Austausch eines (kurzen) Passworts. In der dritten Variante liegt die Verantwortung beim Nutzer, durch die Geräte angezeigte Werte miteinander zu vergleichen.

Da die Übersetzung von Einweg-Pins bei der indirekten Authentifizierung im Vordergrund steht, wird fortan nur noch die zweite Variante von Hos Protokollgruppe betrachtet. Außerdem ist die erste Variante nur vor Einsatz im Feld praktisch durchführbar, sofern alle potentiellen Kommunikationspartner vorher bekannt sind. Die dritte Variante birgt das Rushing-User-Problem, das im Verlauf dieses Kapitels diskutiert wird.

Abbildung 4.2 zeigt den Ablauf des Elliptic Curve Diffie Hellman (ECDH) Protokolls. Zunächst einigen sich beide Parteien öffentlich auf den Parameter G, der einer elliptischen Kurve entspricht. Anschließend wählen beide Parteien ihren geheimen Schlüssel SK_A bzw. SK_B und berechnen aus diesen ihre öffentlichen Schlüssel $PK_A = SK_A \times G$ bzw. $PK_B = SK_B \times G$. Die öffentlichen Schlüssel können nun ungesichert übertragen werden, da die Umkehroperation zur Berechnung der geheimen Schlüssel aus den öffentlichen Parametern mit praktischen Mitteln nicht möglich ist. Den gemeinsamen Sitzungsschlüssel können nun beide Parteien aus dem eigenen privaten Schlüssel und dem öffentlichen Schlüssel des Gegenübers berechnen.

Um einen MITM-Angriff durch den Austausch eines Passworts zu verhindern, ist die Übertragung weiterer Parameter sowie ein zusätzlicher Kommunikationsschritt notwendig. Abbildung 4.3 zeigt den vollständigen Ablauf. Die Schritte des klassischen ECDH-Protokolls sind dabei grau markiert. Zunächst muss das gemeinsame Passwort PW out of band ausgetauscht werden, danach legen beide Parteien ihre geheimen und öffentlichen ECDH-Parameter (Schlüs-

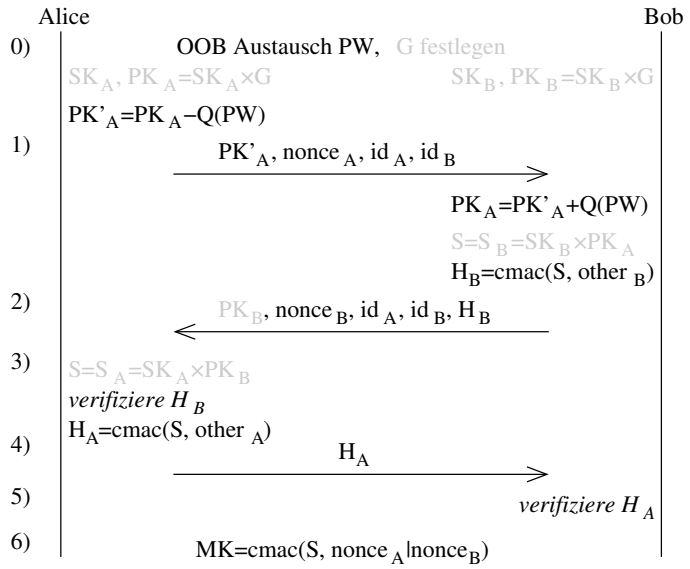


Abbildung 4.3: Ablauf des Protokolls nach Ho mit Authentifizierung über Passwort

sel und Kurvenform) fest. Der beginnende Partner – hier Alice – überträgt nun nicht wie in Schritt 1) in Abb. 4.2 seinen öffentlichen Schlüssel sondern verschlüsselt ihn zunächst mit PW. Hierzu berechnet Alice einen Punkt Q in Abhängigkeit von PW, sodass Q einen Punkt auf der elliptischen Kurve G darstellt. Dieser kann nun so vom öffentlichen Schlüssel PK_A subtrahiert werden, wodurch

$$PK'_A = PK_A - Q(PW) \tag{4.1}$$

entsteht. PK'_A ist dabei ein gültiger Punkt auf der Kurve G, ein Angreifer kann also nicht erkennen, dass es sich dabei nicht um den echten öffentlichen Schlüssel von Alice handelt. PK'_A wird nun in Schritt 1) zusammen mit einer Zufallszahl $nonce_A$ und den Identifikationen von Alice und Bob id_A und id_B übertragen. Bei letzteren kann es sich z.B. um ihre IP- oder Hardware-Adressen handeln.

Da Bob im Besitz von PW ist, kann er den eigentlichen öffentlichen Schlüssel von Alice berechnen:

$$PK_A = PK'_A + Q(PW) \tag{4.2}$$

Dadurch kann Bob das gemeinsame Geheimnis S aus seinem geheimen und Alices öffentlichem Schlüssel berechnen. Um seine Authentizität zu bestätigen,

berechnet Bob einen keyed-Hash H_B aus den öffentlichen Parametern und nutzt dabei S als Schlüssel. In Schritt 2) überträgt Bob seinen öffentlichen Schlüssel, eine weitere Zufallszahl, die Identifikatoren sowie den berechneten Hash an Alice.

Mit dem empfangenen öffentlichen Schlüssel von Bob kann Alice das Geheimnis S berechnen und Bobs Hash verifizieren. Diese Verifikation gelingt nur, wenn H_B mit dem gleichen Geheimnis S gebildet und überprüft wurde. Dieses Vorgehen entspricht demnach einem Challenge-Response-Verfahren. Gelingt die Verifikation, ist Bobs Identität bestätigt. Außerdem werden die öffentlichen Parameter durch ihre Einbeziehung in die Hash-Funktion geschützt, wodurch sie nicht im Transit ausgetauscht werden können, da das ebenfalls zu einem ungültigen Hash-Wert führen würde.

Alice authentifiziert sich auf die selbe Weise gegenüber Bob. Anschließend berechnen Alice und Bob den Master Key MK aus den ausgetauschten Zufallszahlen und dem Geheimnis S. Der Master Key ist das finale Schlüsselmaterial, aus dem nun Schlüssel für Signaturbildung und Verschlüsselung abgeleitet werden können.

4.2.2 Anpassungen am Protokoll von Ho

Nun soll der Fall betrachtet werden, dass eine dritte Instanz die Übersetzung des OOB ausgetauschten Passworts übernimmt und die übrigen Nachrichten transparent weiterleiten soll. In diesem Fall besteht ein Problem bei der Bildung der Hash-Werte H_A und H_B, da diese die öffentlichen Parameter beinhalten. Eine dritte Instanz muss in der Lage sein, diese Parameter anzupassen, um die indirekte Authentifizierung so transparent wie möglich durchzuführen. Zur Bildung der Hash-Werte ist jedoch die Kenntnis des gemeinsamen Geheimnisses S von Alice und Bob notwendig. Diese Kenntnis kann ein Vermittler jedoch nicht erlangen, da er dafür mindestens einen der geheimen Schlüssel bräuchte.

Aus diesem Grund wird in dieser Arbeit eine Änderung am Protokoll von Ho vorgeschlagen (vgl. Abb. 4.4). Anstelle des von PW abhängenden gemeinsamen Geheimnisses S wird nun PW direkt als Schlüssel zur Bildung der Hash-Werte H_A und H_B genutzt. Darüber hinaus werden die Hash-Werte zusätzlich über die jeweiligen öffentlichen Schlüssel gebildet. Die Auswirkungen auf die Sicherheit des Protokolls werden im folgenden Abschnitt diskutiert.

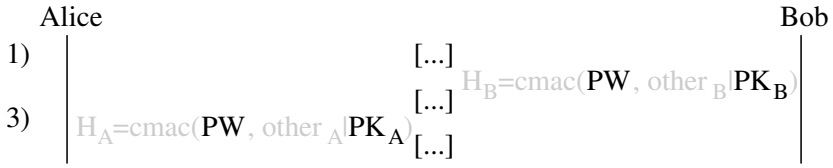


Abbildung 4.4: Vorgeschlagene Änderungen am ECDH-Handshake nach Ho ([Ho12])

4.2.3 Sicherheitsbetrachtung

Die Authentifizierung zwischen Alice und Bob gelingt nur, wenn beide identische gemeinsame Geheimnisse berechnen.

$$S_A = S_B = S \tag{4.3}$$

In Hos Protokoll hängt S und damit auch H_B dabei von PW ab:

$$H_B = f(S) \tag{4.4}$$
$$S = f(PK_A) \tag{4.5}$$
$$PK_A = f(Q) \tag{4.6}$$
$$Q = f(PW) \tag{4.7}$$
$$\rightarrow H_B = f(f(f(f(PW)))) \tag{4.8}$$

Ein MITM-Angreifer, der um das gesamte Protokoll und die damit einhergehende Verschlüsselung weiß, muss demnach einen Brute-Force-Angriff, also ein systematisches Durchprobieren aller Werte, über den gesamten Schlüsselraum von PW durchführen. Es sei an dieser Stelle erwähnt, dass ein vergleichsweise kleiner Schlüsselraum für PW ausreichend ist, da seine Geheimhaltung nur für den Vorgang der Authentifizierung erforderlich ist. Ist diese ein mal abgeschlossen, können aus PW keine Rückschlüsse auf S und damit auf den Masterkey MK gezogen werden.

Durch die erste in dieser Arbeit vorgeschlagene Änderung, die Nutzung von PW als Schlüssel für die keyed-Hash-Funktion, entsteht demnach keine Beeinträchtigung für die Sicherheit des Protokolls. H_B hängt nun nicht mehr mittelbar sondern direkt von PW ab:

$$H_B = f(PW) \tag{4.9}$$

Ein Brute-Force-Angriff muss auch in dieser Variante über den gesamten Schlüsselraum von PW erfolgen.

Einem Man-In-The-Middle-Angreifer wäre es möglich, in dem so entstehenden Handshake beispielsweise Alices öffentlichen Schlüssel durch seinen eigenen auszutauschen. Zwar berechnen Alice und Bob unterschiedliche Sitzungsschlüssel, der Angreifer als solcher würde jedoch unerkannt bleiben. Es wäre lediglich keine Kommunikation zwischen Alice und Bob möglich. Die vier in Kapitel 2.2.1 genannten Schutzziele sind zwar dadurch nicht verletzt, es beeinträchtigt jedoch die Zuverlässigkeit des Gesamtsystems, da keiner der Teilnehmer erkennt, dass eine Authentifizierung fehlgeschlagen ist.

Um diese Angriffsmöglichkeit zu unterbinden, berücksichtigen beide Teilnehmer bei der Bildung ihrer Hashes H_A und H_B ihre eigenen öffentlichen Schlüssel. Bei dem skizzierten Angriff schlägt nun die Verifikation eines der Hash-Werte fehl und die Authentifizierung wird als gescheitert betrachtet. Das vollständige Protokoll ist in Abbildung 4.5 dargestellt.

4.3 Umsetzung als WS4D Mobile Authenticator

Um die Machbarkeit einer indirekten Authentifizierung – also dem Übersetzen von OOB-Kanälen – mit dem modifizierten ECDH-Protokoll nach Ho zu demonstrieren, wurden in dieser Arbeit zunächst das zugrundeliegende kryptografische Protokoll aus Abschnitt 4.2.2 prototypisch implementiert und darauf basierend ein praktisch einsetzbares Kommunikationsprotokoll auf Basis des Devices Profile for Web Services (DPWS) entworfen. In diesem Abschnitt werden das entworfene Protokoll sowie die Hard- und Softwarekomponenten der gesamten prototypischen Umsetzung beschrieben.

4.3.1 Protokoll zur indirekten Authentifizierung

Auf Basis des modifizierten ECDH-Handshakes nach Ho wurde zur Umsetzung des Prototyps die konkrete Inkarnation eines Kommunikationsprotokolls entworfen, die in Abbildung 4.6 zu sehen ist.

Vorausgesetzt wird hier, dass ein Client C bereits ein Device D ausgewählt hat, mit dem es sich authentifizieren möchte. Ferner wird angenommen, dass ein Gerät verfügbar ist, das vom Typ `<ws4d:UI-Authentictor>` ist (vgl.

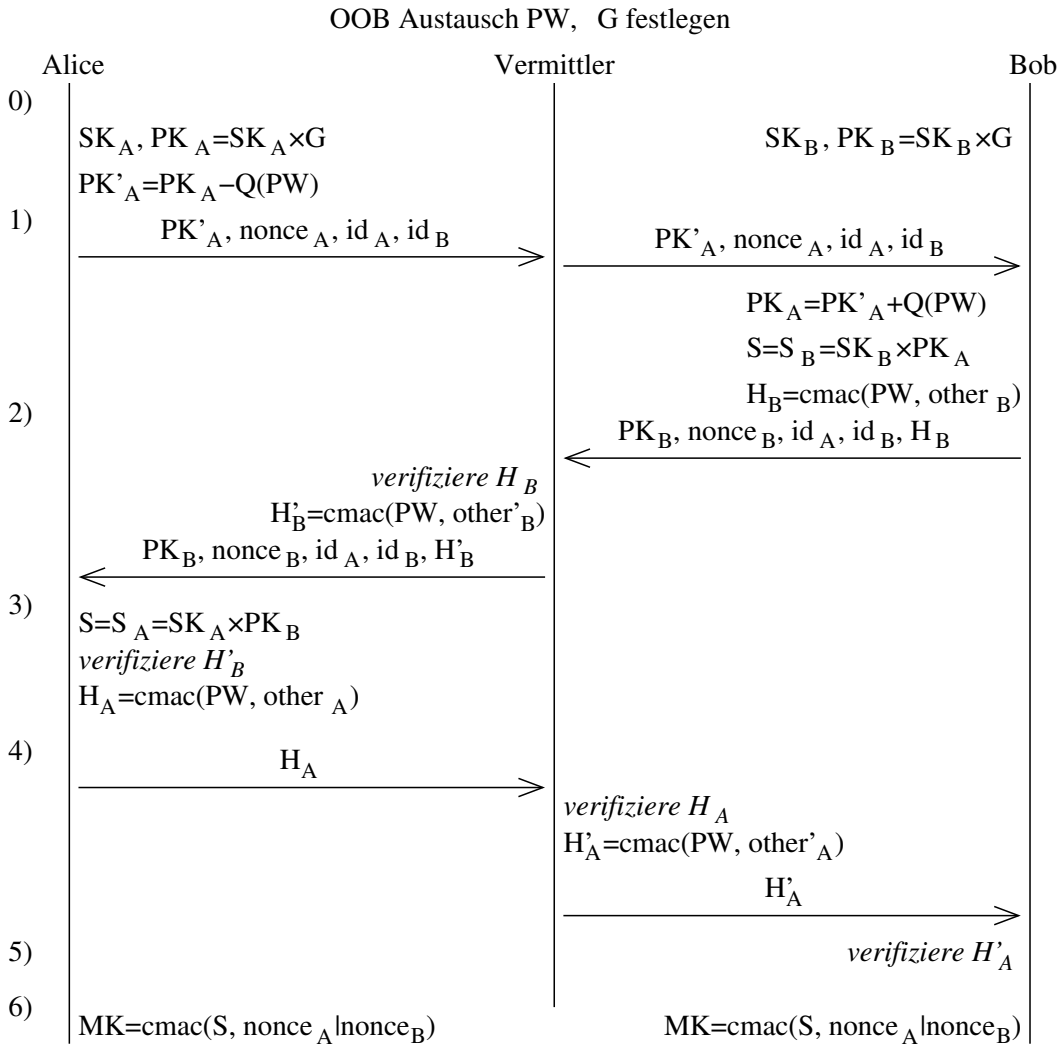


Abbildung 4.5: Ablauf des modifizierten Protokolls nach Ho über einen Mittelsmann

Anhang B.9) (UI in der Abbildung) und damit als der hier beschriebene Vermittler fungieren kann. Demnach hat es einen such- und auffindbaren Authentifizierungsdienst definierten Typs, dessen Beschreibung die unterstützten OOB-Authentifizierungsmethoden unterstützt. Das Device besitzt einen Dienst mit identischer Schnittstelle, ist jedoch vom Typ `<ws4d:AuthentictionEndpoint>`. Damit wird signalisiert, dass eine direkte Authentifizierung mit ihm möglich ist. Für den einfachsten Fall wird angenommen, dass eine indirekte Authentifizierung möglich ist, dass also UI je einen Mechanismus unterstützt, die von Client und Device benötigt werden. Eine Lösung für den Fall, dass diese Annahme nicht zutrifft wird in Abschnitt 4.5.1 diskutiert.

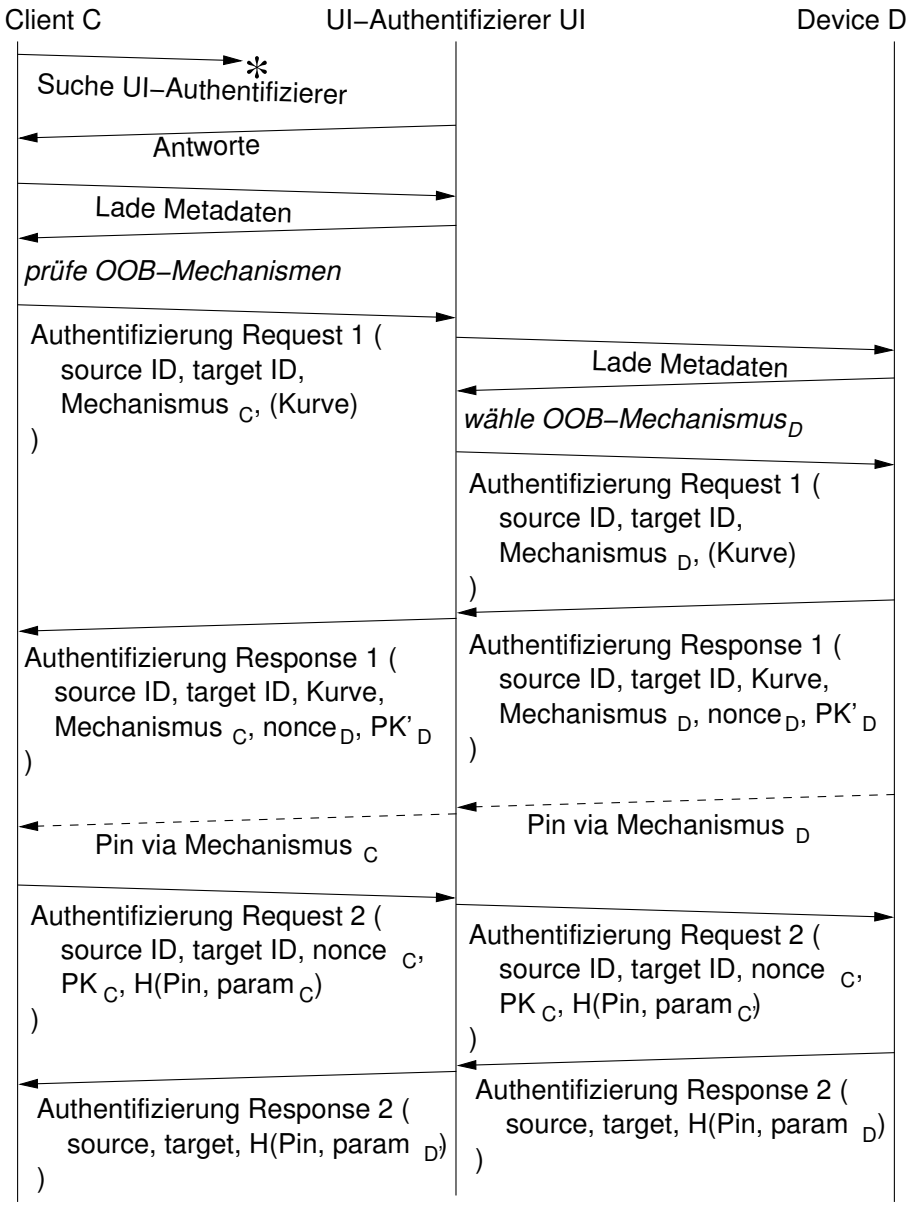


Abbildung 4.6: Ablauf des vollständigen Protokolls zur indirekten Authentifizierung

Nachdem der Client das Device und den UI-Authentifizierer aufgefunden hat, lädt er die Beschreibung des Authentifizierungsdienstes von UI herunter und wählt einen OOB-Authentifizierungsmechanismus aus. Anschließend richtet er einen Request an UI und beantragt eine Authentifizierung mit D. Dieser Request beinhaltet Identifikatoren für Client und Device, den Bezeichner für den ausgewählten Mechanismus (s. Anhang B.9) und optional den Bezeichner für die gewünschte elliptische Kurve.

UI überprüft, ob es den von C geforderten Mechanismus unterstützt. Im Erfolgsfall lädt UI die Beschreibung des Authentifizierungsdienstes des Devices herunter und wählt einen gemeinsamen OOB-Authentifizierungsmechanismus aus. Anschließend kann der ursprüngliche Request von C an D weitergeleitet werden, jedoch mit ausgetauschtem OOB-Authentifizierungsmechanismus.

Das Device überprüft seinerseits den angeforderten Mechanismus und wählt im Erfolgsfall eine Pin aus, mit der es seinen öffentlichen Schlüssel zu PK'_D verschlüsselt. Bei der Wahl des Schlüsselpaars berücksichtigt D eine eventuelle Präferenz hinsichtlich der elliptischen Kurve von C. Das Device antwortet mit den Identifikatoren, dem Mechanismus, dem Kurvennamen, einer Nonce und seinem verschlüsselten öffentlichen Schlüssel. Der UI-Authentifizierer leitet die Antwort weiter und tauscht dabei lediglich den Mechanismus aus.

Das Device beginnt nach Versenden seiner Antwort damit, die gewählte Pin über den von ihm gewählten OOB Kanal zu versenden bzw. anzuzeigen. Der Client hingegen befindet sich nach Erhalt der ersten Antwort in einem Zustand, in dem er die Pin empfangen bzw. entgegennehmen kann. Der UI-Authentifizierer ist schließlich dafür zuständig, die PIN über einen OOB-Kanal von D entgegenzunehmen, zu übersetzen und schließlich über einen anderen Kanal an C zu übermitteln.

Nach Empfang der Pin stellt der Client einen Folgerequest an UI. Dieser enthält unverändert die Identifikatoren zum Zwecke der Zuordnung des Requests. Außerdem überträgt er seine Zufallszahl, seinen öffentlichen Schlüssel und den keyed-Hash seiner Parameter, wobei die Pin als Schlüssel dient. Die gehashten Parameter beinhalten die Identifikatoren, den gewählten Mechanismus, den Namen der genutzten Kurve, die Zufallszahlen und seinen öffentlichen Schlüssel. Um die Daten wie gewohnt weiterleiten zu können, muss UI den Mechanismus austauschen und den Hash-Wert aufgrund des geänderten Mechanismus neu berechnen. Zuvor überprüft UI den empfangenen Hash-Wert und antwortet bei Misserfolg mit einer Fehlermeldung.

Das Device überprüft den Hashwert, und antwortet auf die gleiche Weise. Konnten die Hash-Werte auf beiden Seiten verifiziert werden und hat UI keinen Fehler festgestellt, ist die Authentifizierung geglückt. Beide Seiten berechnen das gemeinsame Geheimnis S und daraus abschließend den Master Key MK.

Abbildung 4.7: Hardware-Komponenten des Prototyps, v.l.n.r netzwerkfähiger Lichtschalter, netzwerkfähige Lampe und LG Nexus 4 exemplarisch als Multimediagerät

4.3.2 Hardware Prototypen

Zur prototypischen Umsetzung der indirekten Authentifizierung kommen drei verschiedene Geräte zum Einsatz (vgl. Abbbildung 4.7). Als UI-Authentifizierer, der die Übersetzung der OOB-Kanäle vornimmt, kam ein Android-Smartphone zum Einsatz. Je nach Entwicklungsstand handelte es sich um ein Samsung Nexus S (Android 4.1.2, ARM Cortex A8 CPU mit 1GHz, 512MByte RAM) bzw. um ein LG Nexus 4 (Android 4.1.2 – Android 4.4.4, Quad-Core ARM Cortex A9 CPU mit 1,5 GHz, 2GByte RAM).

Als Device wurde eine netzwerkfähige Lampe entwickelt, der zugehörige Client ist ein netzwerkfähiger Lichtschalter. Als Recheneinheit kommt bei beiden ein Raspberry Pi Modell B zum Einsatz. Dabei handelt es sich um einen kreditkartengroßen Einplatinencomputer mit einer BCM2835 ARM11-CPU von Broadcom, die auf 700MHz getaktet ist und 512MB Arbeitsspeicher. Über eine Pin-Leiste sind 8 digitale GPIOs nutzbar. [RS 13]

Um den Taster auszulesen und die Lampe zu steuern, mussten sie an einen GPIO des jeweiligen Raspberry Pi angeschlossen werden, wozu zusätzliche schaltungstechnische Maßnahmen notwendig sind. Beim Anschluss des Tasters genügt es, den Eingang durch Schutzdiode und -widerstand gegen zu hohe Eingangsströme bzw. -spannungen zu sichern und über einen Pull-Down-Widerstand ein definiertes Potential bei geöffnetem Taster zu erzeugen.

Sind die GPIOs hingegen als Ausgang beschaltet, so sind sie in ihrer Grundkonfiguration mit 8mA belastbar. Um das verbaute Relais zu betreiben, werden

allerdings 72mA benötigt ([FIN11]), wobei hohe Schaltströme noch nicht berücksichtigt wurden. Daher wurde eine Treiberstufe entworfen und eingefügt. Zusätzlich schützt eine Zenerdiode den GPIO vor Überspannungen. Sowohl die Eingangs- als auch die Ausgangsbeschaltung wurden als Module für das Raspberry Pi angefertigt. Die Schaltpläne sind in Anhang C zu finden.

Sowohl die Lampe als auch der Schalter haben keinerlei zusätzliche Peripherie verbaut wie z.B. Displays oder Tastaturen. Demnach ist die einzige Möglichkeit der Lampe zur Ausgabe, die Lampe ein- oder auszuschalten. Entsprechend verfügt der Lichtschalter einzig über die Eingabemöglichkeit, dass der Schalter selbst gedrückt ist oder nicht. Die Smartphones hingegen besitzen eine Vielzahl an Schnittstellen. Unter ihnen befinden sich jeweils ein hochauflösendes Farbdisplay mit Touchscreen, vollwertige Tastaturen, zwei Fotokameras (die auch als Infrarotempfänger dienen können), eine Blitz-LED, Helligkeitssensoren, Nahfeldkommunikation (NFC, nur LG Nexus 4), Mikrofon und Lautsprecher, Beschleunigungssensoren und viele mehr.

4.3.3 Softwarekomponenten

Die eingesetzten Raspberry Pis zeichnen sich dadurch aus, dass auf ihnen Linux (hier „Raspbian Wheezy„) als Betriebssystem zum Einsatz kommt. Ebenfalls für das Raspberry Pi erhältlich ist die Java-Laufzeitumgebung OpenJDK. Auf den eingesetzten Smartphones läuft das Betriebssystem Android, das auf Linux basiert. Mit der Dalvik VM steht hier ebenfalls eine Java-Laufzeitumgebung zur Verfügung. Hierdurch entstand ein prototypisches Setup mit einer gemeinsamen Basis an Betriebssystem, Laufzeitumgebung und Programmiersprache (Java). Die gesamte Software-Architektur des Prototyps ist in Abbildung 4.8 dargestellt, die zur Umsetzung notwendigen Lines of Code in Tabelle 4.1.

Auf allen Komponenten wird das DPWS-Framework JMEDS aus dem Entwickler-SVN genutzt, wodurch stets die aktuellste Version zur Verfügung steht. Die kryptografische Funktionalität wird durch die offene Lightweight-Cryptography-Bibliothek Bouncy Castle in Version 1.49 bereitgestellt ([Leg14]). Da es unter Android bei der Verwendung von Bouncy Castle zu Problemen kommen kann, existiert das Derivat Spongy Castle, das hier in Version 1.47 eingesetzt wird ([Git13]). Auf Basis der Krypto-Bibliotheken wurde für diese Arbeit eine Wrapper-Bibliothek entworfen und entwickelt, die die kryptografi-

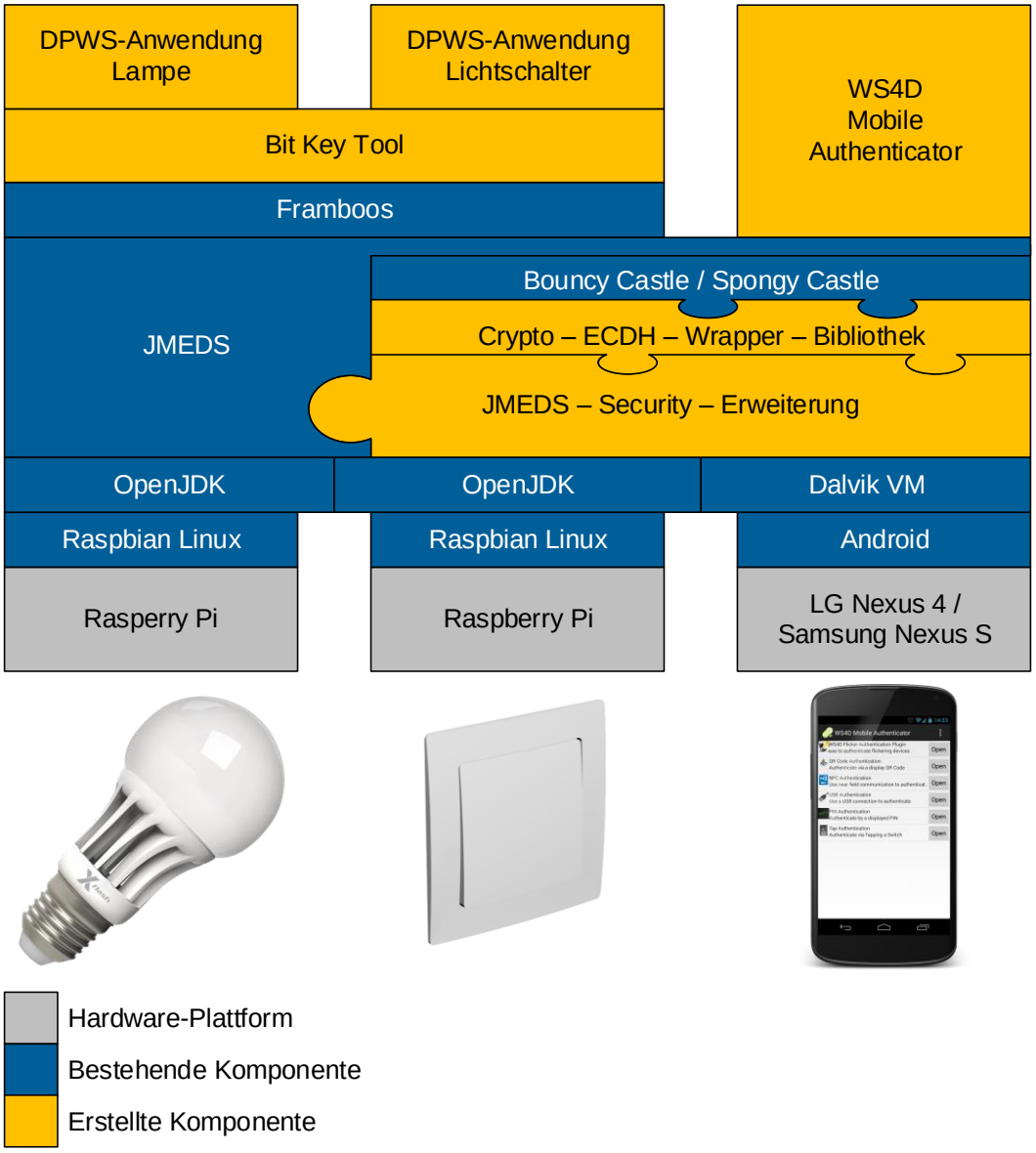


Abbildung 4.8: Software-Architektur der prototypischen Umsetzung

schen Komponenten des in Abschnitt 4.3.1 beschrieben Protokolls kapselt und in einer komfortablen API zur Verfügung stellt.

Die Bibliothek framboos ([Kra14]) stellt eine bequeme Möglichkeit dar, die GPIOs des Raspberry Pi über eine Java-API zu steuern und auszulesen. Sie wird in der Entwicklerversion vom 07.06.2013 genutzt, zusammen mit Verbesserungen die der Autor dieser Arbeit vorgenommen hat und am 08.08.2013 offiziell Einzug in den Quellcode von framboos hielten. Auf Basis dieser Bibliothek wurde im Rahmen dieser Arbeit eine Möglichkeit geschaffen, numerische

Komponente	Lines of Code
Erweiterungen JMEDS	4365
Crypto-Wrapper	1225
Bit Key Tool	132
Anwendung Lampe	541
Anwendung Schalter	1227
WS4D Mobile Authenticator App	6488
Test- & Debugging-Software	269
Gesamt	**14247**

Tabelle 4.1: Auflistung der Umfänge der Prototypkomponenten

Pins in ihrer binären Form über die GPIOs eines Raspberry Pis ein und auszugeben. Auf dieser Basis kann die Lampe den Wert einer Pin flackern, der Lichtschalter hat die Möglichkeit der Eingabe einer Pin.

Auf Basis all dieser Komponenten wurden zunächst Lampe und Lichtschalter entwickelt. Beide werden durch eine Java-Anwendung repräsentiert und kommunizieren über DPWS. Die Lampe bietet zwei Dienste an. Neben dem Dienst zur (in)direkten Authentifizierung wie in Abschnitt 4.3.1 bzw. in Anhang B.9 beschrieben, bietet ein weiterer Dienst zwei Operationen an um den derzeitigen Status der Lampe (an oder aus) zu erfragen und um einen neuen Status zu setzen, die Lampe also ein- oder auszuschalten. Die Beschreibung des Authentifizierungsdienstes enthält zudem einen Policy-Abschnitt, in dem festgelegt wird, dass der einzig unterstützte Mechanismus zum OOB Austausch einer Pin der ist, diese PIN zu flackern.

Der Lichtschalter wird durch die zugehörige Client-Anwendung dargestellt. Nach dem erstmaligen Einschalten – also vor der Authentifizierung – sucht er eine passende Lampe sowie vorhandene UI-Authentifizierer, von denen er den ersten auswählt, der seinen Mechanismus, das „Eintastern“ eines Schlüssels, unterstützt. Wird der Taster nun betätigt, gibt der Client der Lampe den Befehl zu schalten. Dieser Befehl wird von der Lampe wegen des fehlenden Vertrauensverhältnisses mit einem Fehler quittiert, woraufhin der Client das in Abschnitt 4.3.1 beschriebene Protokoll einleiten wird. Gelingt die Authenti-

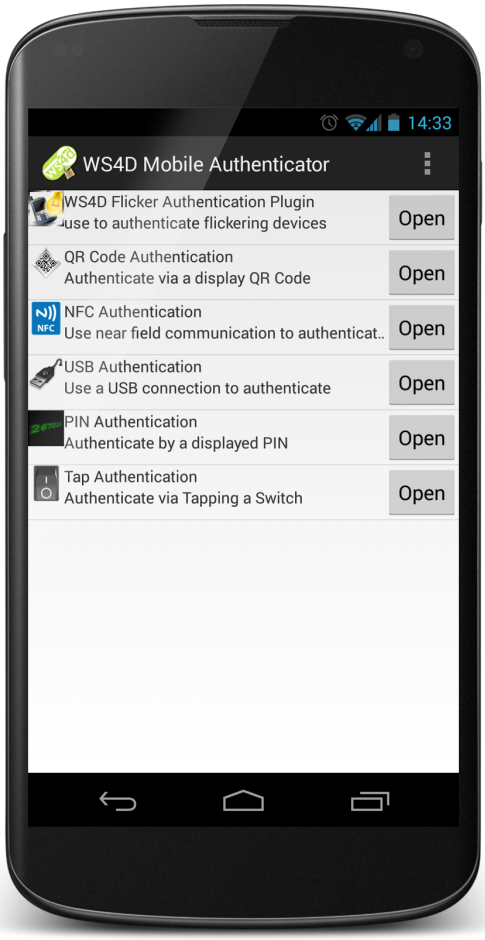


Abbildung 4.9: Ansicht der WS4D Mobile Authenticator App für Android nach dem Start

fizierung, wird auf das nächste Betätigen des Schalters das Schalten der Lampe folgen.

Das Softwarekernstück des Prototypen für die indirekte Authentifizierung stellt die *WS4D Mobile Authenticator* Android App dar. Ihre Aufgabe ist es, zwischen Client und Device zu vermitteln und die OOB-Kanäle ineinander zu übersetzen. Startet man die App, präsentiert sie sich als eine Liste verfügbarer Plug-ins (Abbildung 4.9). Ein Plug-in repräsentiert hierbei die Unterstützung eines OOB-Übertragungsmechanismus wie z.B. sichtbares Licht, NFC oder eine USB-Verbindung.

Für die prototypische Umsetzung wurden in dieser Arbeit zunächst drei Plug-ins implementiert (s. Abbildung 4.10). Das Plug-in zur Flackerauthentifizierung wertet einen kleinen Bereich einer Kameraansicht aus und errechnet

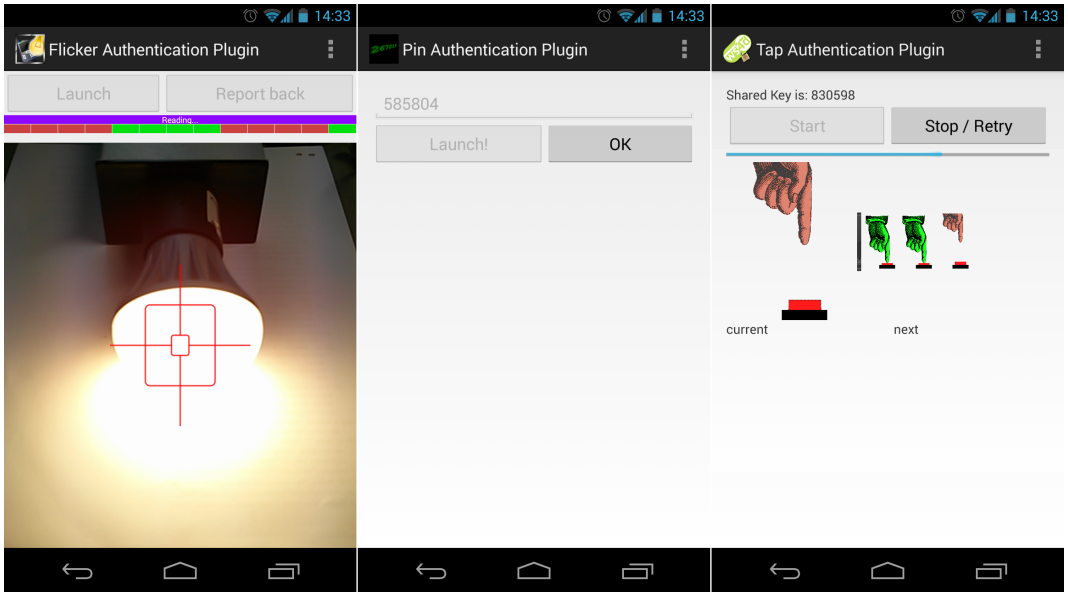


Abbildung 4.10: Implementierte Plug-ins für den WS4D Mobile Authenticator: Übertragung durch Flackern, als numerische Pin und als Muster zum „Eintastern“ (v.l.n.r)

daraus einen Helligkeitswert. Das Plug-in unterscheidet dadurch die Zustände „hell“ und „dunkel“, wodurch die Übertragung einer Pin in binärer Repräsentation durch sichtbares Licht möglich ist. Um die eingefangene Pin in den Lichtschalter „eintastern“ zu können, stellt ein zweites Plug-in sie als eine Sequenz aus Befehlen dar, den Taster zu betätigen oder nicht. Außerdem wurde ein Plug-in entwickelt, dass Pins in ihrer numerischen Form darstellt und entgegennimmt. Dies fand Anwendung in einer Erweiterung, die in Abschnitt 4.5.1 besprochen wird und zu Debugging-Zwecken.

Die Funktionsweise der App wird am konkreten Beispiel von Lichtschalter und Lampe besprochen (vgl. Abbildung 4.11). Nach den beschriebenen Discovery-Vorgängen und dem Bezug der Beschreibung richtet der Schalter den Request zur Authentifizierung mit der Lampe an die DPWS-Device-Logik der App (1)), die von einem Android-Hintergrunddienst verwaltet wird. Dieser lädt die Dienstbeschreibung der Lampe herunter, wählt einen geeigneten Mechanismus (Flackern) und bestimmt das passende Plug-in (2)). Da es sich um ein asynchron eintreffendes Ereignis handelt, das eine Nutzerinteraktion verlangt, wird die Benachrichtigungsfunktion von Android genutzt, um den Nutzer auf das Ereignis hinzuweisen (3)). Nach der Bestätigung durch den

Nutzer wird das passende Plug-in geöffnet (4)), das sich einerseits dafür bereit macht, die PIN OOB entgegenzunehmen und außerdem die Client-Logik anweist, eine Authentifizierung mit der Lampe vorzunehmen (5)).

Das Device verarbeitet und beantwortet den Request. Die Client-Logik informiert die Device-Logik asynchron über das Eintreffen der Antwort (7a)), sodass der ursprüngliche Request des Schalters ebenfalls beantwortet werden kann (8a)) und dieser sich für den OOB Empfang der Pin bereit macht. Zeitgleich mit der asynchronen Verarbeitung der Antwort wird synchron die Pin vom bereits gestarteten Plug-in entgegengenommen (6)).

Das Plug-in informiert den Programmkern über das Eintreffen einer Pin zusammen mit dem Hinweis, auf welche Weise sie nun ausgegeben werden soll (7b)). Im Falle des Beispiels soll die Pin demnach als Tapping-Muster dargestellt werden, weswegen das entsprechende Plug-in geöffnet wird (8b)). Durch diese Anzeige kann die Pin nun vom Client eingelesen werden (9)). Ist das geschehen, wird der Handshake fortgesetzt (10)). Das erfordert keine weitere Nutzerinteraktion und passiert synchron im Hintergrund.

Die besondere Herausforderung beim Entwurf dieser Architektur lag in der Tatsache, dass die eigentlich synchrone Kommunikation zu asynchronen Nutzerinteraktionen führt, auf die das System reagieren muss. Der synchrone Aufruf der Device-Logik kann nicht einfach zu einem synchronen Client-Aufruf führen. Stattdessen mussten vergleichsweise komplexe interne Benachrichtigungssysteme aufgebaut und genutzt werden, um die asynchronen Ereignisse wie z.B. Nutzerinteraktion oder Eintreffen einer OOB-Pin mit der eigentlichen Kommunikation zu synchronisieren.

4.4 Abläufe für den Nutzer

Zur Veranschaulichung des Protokolls und des Zusammenspiels der Komponenten wird nun der gesamte Ablauf der indirekten Authentifizierung aus Sicht des Nutzer am Beispiel des aufgebauten Prototyps beschrieben.

Betätigt der Nutzer den Schalter, prüft dieser Vorhandensein von Lampe und UI-Authentifizierer und richtet den ersten Authentifizierungs-Request an das Smartphone. Der Nutzer wird darüber vom Android-Benachrichtigungssystem informiert, wodurch er erfährt, welche Geräte sich miteinander au-

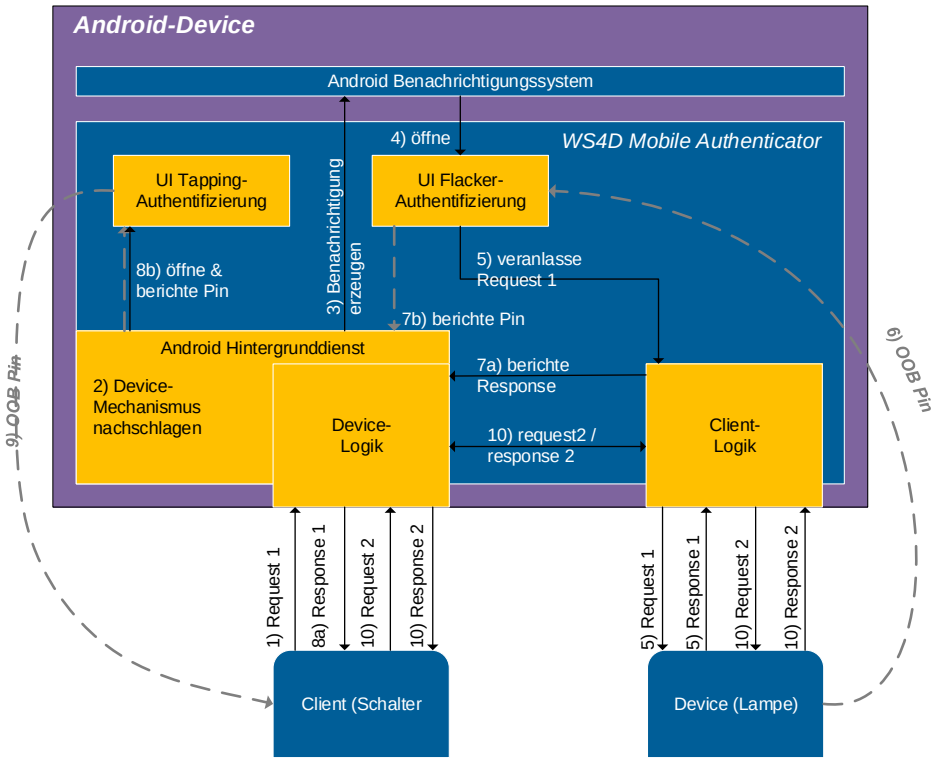


Abbildung 4.11: Architektur der WS4D Mobile Authenticator App

thentifizieren wollen und welche Mechanismen dabei eingesetzt werden sollen (Abb. 4.12-1).

Bestätigt der Nutzer die Benachrichtigung, öffnet sich die Nutzeroberfläche für die Flacker-Authentifizierung. Der Nutzer wird aufgefordert, ein Fadenkreuz auf die Lampe zu richten und dies zu bestätigen. Die Bestätigung leitet im Hintergrund den ersten Request an die Lampe weiter, die nach dessen Beantwortung beginnt, die generierte Pin zu flackern (Abb. 4.12-2).

Das Ende des Flackerns muss ebenfalls durch den Nutzer bestätigt werden, wodurch er zur Plug-in-Übersicht geleitet wird. Hier weist ein auffälliger Button auf eine aktive Authentifizierung hin (Abb. 4.12-3). Wird dieser betätigt, öffnet sich die Nutzeroberfläche des Tapping-Plug-ins und der Nutzer kann die Wiedergabe der Sequenz zum „Eintastern“ der Pin beginnen (Abb. 4.12-4). Ist das abgeschlossen, sind die Vorgänge aus Sicht des Nutzers beendet.

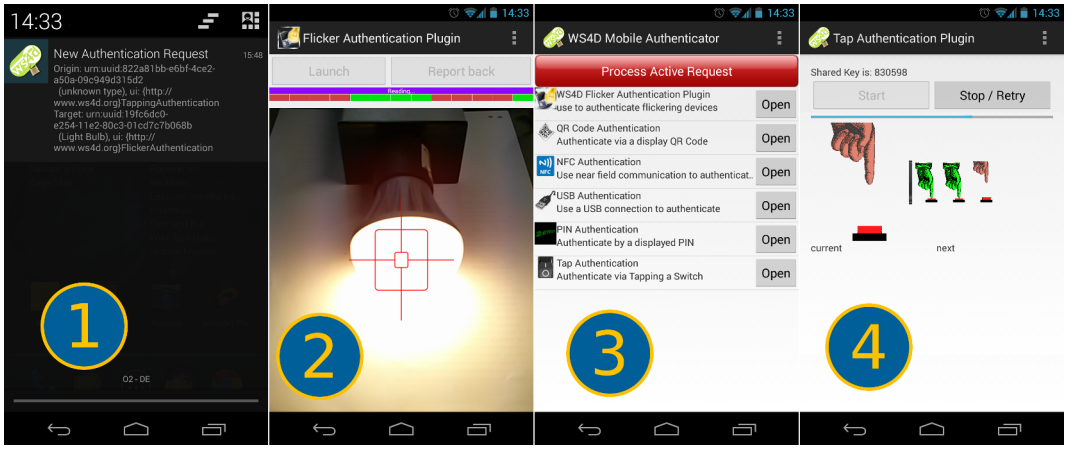


Abbildung 4.12: Ablauf einer indirekten Authentifizierung für den Nutzer am Beispiel

4.5 Erweiterungen des Protokolls

In diesem Abschnitt werden zwei Erweiterungen der indirekten Authentifizierungen besprochen, die die Vielseitigkeit erhöhen und die Anwendung vereinfachen sollen. Beide Erweiterungen wurden im bestehenden Prototypen implementiert und somit ihre Machbarkeit nachgewiesen.

4.5.1 Kaskadierung von Multimediageräten

Bei der Diskussion des entwickelten Protokolls zur indirekten Authentifizierung in Abschnitt 4.3.1 wurde angenommen, dass ein Gerät von Typ `<ws4d:UI-Authenticator>` existiert, das je einen OOB-Übertragungsmechanismus für den Client und das Device unterstützt (vgl. Prinzipbild in Abbildung 4.13 a)). Das entwickelte Protokoll zur indirekten Authentifizierung wurde von vornherein so konzipiert, dass es dieses Problem lösen kann, in dem es mehrere Multimediageräte kaskadiert. Somit ist nur noch nötig, dass eine Gruppe von Multimediageräten existiert, die gemeinsam die notwendige Kombination aus OOB-Mechanismen erfüllt. (vgl. Prinzipbild in Abbildung 4.13 b)).

Nach Eingang des ersten Requests vom Client an einen UI-Authentifizierer lädt letzterer die Metadaten vom Ziel der Authentifizierung herunter um einen passenden Mechanismus zu wählen (vgl. Abbildung 4.6). Schlägt das jedoch fehl, verhält sich der UI-Authentifizierer nun ähnlich wie ein Client.

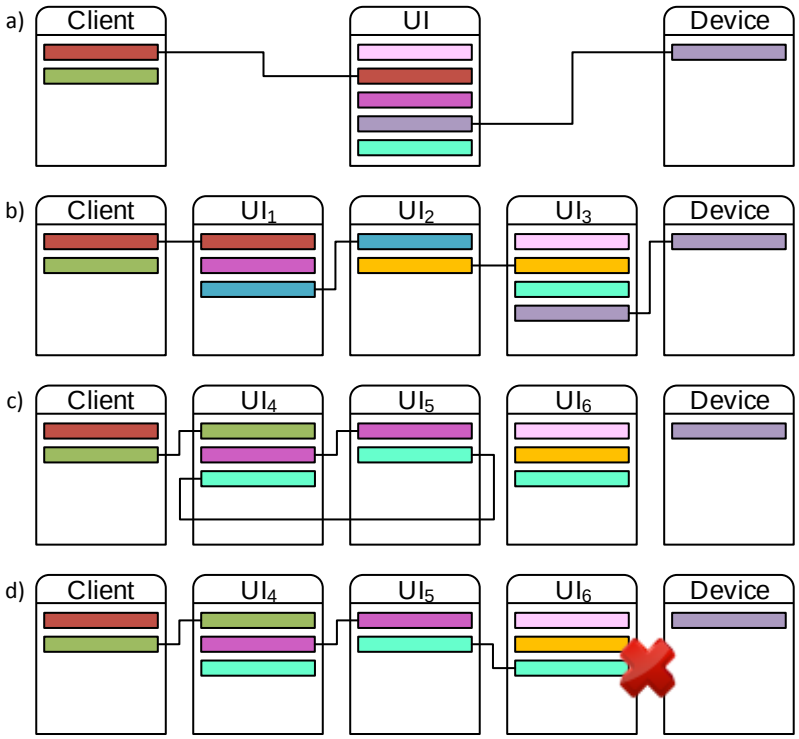


Abbildung 4.13: Prinzipbilder zur Kaskadierung von UI-Authentifizierern
a) Keine Kaskadierung
b) Kaskadierung von drei Authentifizierern
c) Beispiel für einen Zyklus
d) Wahl eines Pfades der nicht zum Erfolg führt

Er sucht seinerseits nach UI-Authentifizierern. Während ein (ressourcenschwacher) Client die Wahl des OOB-Mechanismus des Ziels dem stärkeren Mittelsmann überlässt, versucht ein UI-Authentifizierer bei der Kaskadierung zunächst einen nächsten Vermittler zu finden, der je einen Mechanismus mit dem Ziel und mit sich selbst teilt (s. UI_2 und UI_3 in Abb. 4.13 b)). Nur wenn das fehlschlägt, wählt er den ersten Authentifizierer, mit dem zumindest er allein einen OOB-Kanal aufbauen kann (s. UI_1 und UI_2 in Abb. 4.13 b)). In beiden Fällen wird der ursprüngliche Request an den nächsten UI-Authentifizierer weitergeleitet. Der Unterschied zu einem regulären Request von einem Client besteht einzig darin, dass der Absender des Requests nicht der im Request angegebenen Identifikation der Quelle entspricht. Daran lässt sich erkennen, dass es sich um einen weitergeleiteten Request handelt und nicht um einen initialen. Jeder angesprochene UI-Authentifizierer verhält sich auf die gleiche

Weise, wodurch weitere Kaskdierungsstufen möglich sind, bis ein durchgehender OOB-Kanal aufgebaut wurde.

Bei der Kaskadierung von UI-Authentifizierern können zwei Probleme auftreten. Einerseits sind Zyklen möglich, andererseits ist es denkbar, dass ein Pfad gewählt wird, der nicht zum Erfolg führt (vgl. Abbildung 4.13 c) und d)). Damit die indirekte Authentifizierung eine praktikable Lösung darstellen kann, müssen Strategien für beide Probleme entwickelt werden. Hierbei wird Lösungen Vorzug gegeben, die einerseits keiner Änderungen am Protokoll bedürfen und für die weiterhin kein zusätzliches Wissen notwendig ist.

Zur Vermeidung von Zyklen wird als Verhaltensregel eingeführt, dass UI-Authentifizierer nicht auf Suchanfragen nach ihnen antworten dürfen, während sie sich in einer aktiven Authentifizierungsanfrage befinden. Da das Discovery von UI-Authentifizierern erst in dem Moment durchgeführt wird, wenn sie gebraucht werden, unterbricht dies einen möglichen Zyklus. Im Beispiel in Abbildung 4.13 c) würde UI_4 demnach nicht auf eine Anfrage von UI_5 antworten. Um zu vermeiden, dass sie z.B. durch gecachte Suchanfragen dennoch einbezogen werden, antworten sie mit einem Fehler, falls sie angesprochen werden, während sie sich in einem aktiven Authentifizierungsvorgang befinden.

Um der Wahl eines falschen Pfads zu begegnen, müssen Clients und UI-Authentifizierer alle zur Verfügung stehenden und passenden Vermittler nacheinander ansprechen und dürfen das erst beenden, wenn die Authentifizierung erfolgreich war oder keine weiteren Authentifizierer zur Verfügung stehen. Das verhindert zwar nicht initial das Einschlagen des falschen Pfades, es verhindert jedoch das Fehlschlagen der Authentifizierung, obwohl die Möglichkeit zum Aufbau eines Vertrauensverhältnis bestanden hätte.

4.5.2 Vertraulicher Kanal zwischen Multimediageräten

Die Kaskadierung von Multimediageräten erhöht die Praktikabilität der indirekten Authentifizierung, da durch diese Lösung die Wahrscheinlichkeit erhöht wird, dass ein OOB-Kanal zwischen zwei Geräten zum Austausch einer Einweg-Pin aufgebaut werden kann. Es ist jedoch nicht zu vernachlässigen, dass jede Stufe der Kaskadierung eine Interaktion des Nutzers erfordert. Hinzu kommt die grundsätzliche Fehleranfälligkeit beim OOB-Austausch (Verlesen und Vertippen bei Pin-Eingabe, falsche Ausrichtung des Fadenkreuzes und damit Feh-

lerkennung beim Flackern oder Fehler beim Tapping), wodurch der gesamte Vorgang wiederholt werden muss.

Wie in [SS75] als „Prinzip der psychologischen Akzeptanz“ erwähnt, müssen Sicherheitsmechanismen so unkompliziert und nutzerfreundlich wie möglich ablaufen. Andernfalls neigen Nutzer sonst zu Fehlbedienung oder dazu, Sicherheitsmechanismen vollständig zu deaktivieren. Aus diesem Grund wird der OOB Mechanismus `ws4d:EncryptedPinExchange` eingeführt. Er kann nur zwischen zwei Geräten genutzt werden, die bereits eine Vertrauensbeziehung miteinander aufgebaut haben und in der Lage sind, Nachrichten verschlüsselt auszutauschen. In diesem Fall wird die Pin in ein dafür vorgesehenes Nachrichten-Feld in die erste Antwort eingefügt. Ein UI-Authentifizierer darf eine solche Anfrage nur beantworten, wenn sie von einem vertrauenswürdigen Gerät stammt und verschlüsselt ist und er auf die gleiche Weise verschlüsselt antworten kann.

Das Einbetten der Pin in die erste Response des Authentifizierungs-Handshake ist seiner Definition nach In-Band und demnach nicht Out-of-Band. Es wird nur als OOB-Mechanismus behandelt, um sich nahtlos in das besprochene Schema bei der Kaskadierung von Multimediageräten einzufügen. Die verschlüsselte Übertragung der Pin von einem Gerät zum Anderen geschieht automatisch und reduziert demnach die Zahl der Nutzerinteraktionen um zwei (z.B. Lesen einer Pin an einem Gerät und manuelle Eingabe am anderen). Außerdem genügt dieses Vorgehen der Anforderung an einen OOB-Kanal, kryptografisch sicher zu sein.

4.5.3 Prototypische Implementierung

Um die Machbarkeit der beschriebenen Erweiterungen nachzuweisen, wurden beide in den bestehenden Prototypen integriert. Hierzu waren zunächst grundlegende Änderungen an der WS4D Mobile Authenticator App notwendig.

Zunächst wurde eine Möglichkeit geschaffen, über einen Konfigurationsdialog die angebotenen Authentifizierungsmaßnahmen auszuwählen. So konnte die Software auf zwei verschiedenen Smartphones ausgeführt und eine zusätzliche Lücke in den OOB-Kanälen geschaffen werden. Das eingesetzte Samsung Nexus S bietet in diesem Versuch nur die Authentifizierung über Tapping und numerische Pins an, das LG Nexus 4 nur die Authentifizierung über Flackermuster und numerische Pins (vgl. Abbildung 4.14).

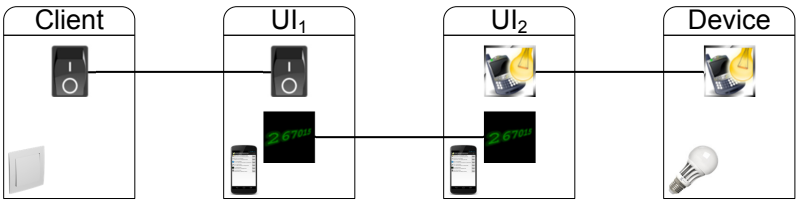


Abbildung 4.14: Schema des Versuchsaufbaus für kaskadierte indirekte Authentifizierung

Da das Nexus 4 kein Tapping unterstützt, wendet sich der Taster nun an das Nexus S. Da dies den einzigen von der Lampe angebotenen Mechanismus (Flackern) nicht unterstützt, sucht es nach einem Gerät, das diesen Mechanismus beherrscht und daneben auch noch einen gemeinsamen Mechanismus anbietet und findet (durch das Anbieten der numerischen Pins) das Nexus 4. Nach der Weiterleitung des Requests öffnet sich auf dem Nexus S ein Feld zur Pin-Eingabe, und das Nexus 4 fängt das von der Lampe geflackerte Muster ein. Danach stellt es die Pin numerisch dar, sodass der Nutzer sie in die Maske des Nexus S eingeben kann, wodurch sie in ein Tapping-Muster übersetzt wird. So kann die Pin zum Schalter gelangen.

Um diesen Ablauf durch eine verschlüsselte In-Band-Übertragung der Pin zwischen den Smartphones zu vereinfachen, wurden diese zunächst um Client- und `ws4d:Endpoint`-Funktionalität ergänzt. Nur so können sie eine direkte Vertrauensbeziehung zueinander herstellen. Nach erfolgter Authentifizierung fordert das Nexus S die Pin vom Nexus 4 nicht mehr als numerische Pin, sondern über den Mechanismus `ws4d:EncryptedPinExchange` an. Im aufgebauten Versuch wird die Anzeige der Pin auf dem Nexus 4 durch einen Hinweis ersetzt, dass der Nutzer die App beenden kann und die Eingabemaske für die Pin auf dem Nexus S wird durch einen Wartebildschirm ersetzt, der verschwindet und die Oberfläche zur Darstellung des Tapping-Muster öffnet, sobald die Pin mit der verschlüsselten Nachricht eintrifft.

4.5.4 Sicherheitsbewertung der Erweiterungen

Die Kaskadierung mehrerer Multimediageräte beeinträchtigt die Sicherheit des Protokolls in soweit, als dass es eine höhere Anzahl von Ansatzpunkten für einen Man-in-the-Middle-Angriff gibt. Sind es bei dem einfachen Protokoll nur

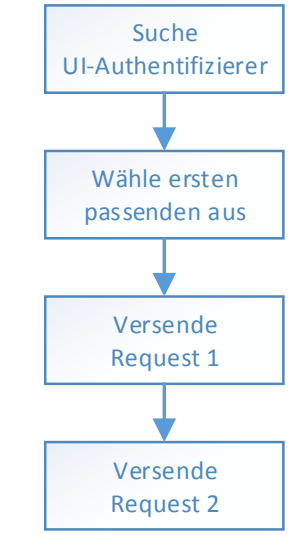
zwei (jeweils zwischen Schalter bzw. Lampe und Smartphone), so kommt mit jedem Multimediagerät ein weiterer hinzu. Um einen MITM-Angriff ausführen zu können, benötigt der Angreifer weiterhin Zugriff auf den OOB-Kanal. Steht dieser jedoch unverändert unter Kontrolle des Nutzers, ist das Protokoll weiterhin als sicher zu betrachten.

Die zweite besprochene Erweiterung ist die Verwendung einer sicheren Kanals zwischen zwei Teilnehmern, der als OOB-Mechanismus betrachtet wird. Dieser kann nur kompromittiert werden, wenn der Aufbau der Vertrauensbeziehung zwischen den beiden Teilnehmern zuvor kompromittiert wurde. Da es sich bei der verschlüsselten In-Band-Übertragung der Pin nicht um ein OOB-Übertragungsverfahren handelt, findet ein MITM-Angriff keine Anwendung.

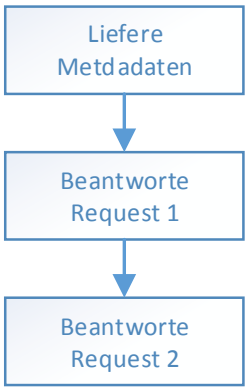
4.6 Zusammenfassung und Diskussion

In diesem Kapitel wird das in dieser Arbeit entwickelte Protokoll zur indirekten Authentifizierung zwischen zwei Geräten besprochen, bei dem ein Multimediagerät die Übersetzung von OOB-Kanälen vornimmt, sodass die Geräte ein Einweg-Passwort austauschen können. Das auf einem authentifizierten Elliptic Curve Diffie Hellman nach [Ho12] basierende Kommunikationsprotokoll berücksichtigt dabei, dass es sich bei den zu authentifizierenden Teilnehmern möglicherweise um ressourcenschwache Geräte handelt und lagert Teile wie die Erkennung der passenden Mechanismen auf die Multimediageräte aus, die zumeist über hinreichende Rechenkraft und Energieversorgung verfügen. In Kombination mit den entwickelten Erweiterungen zur Kaskadierung von Multimediageräten inklusive der verschlüsselten Kommunikation untereinander ergeben sich die in Abbildung 4.15 dargestellten Aufwände. Die vorgeschlagenen Protokolle und Erweiterungen wurden implementiert, der entstandene Prototyp aus Lichtschalter, Lampe und zwei Smartphones wurde erläutert.

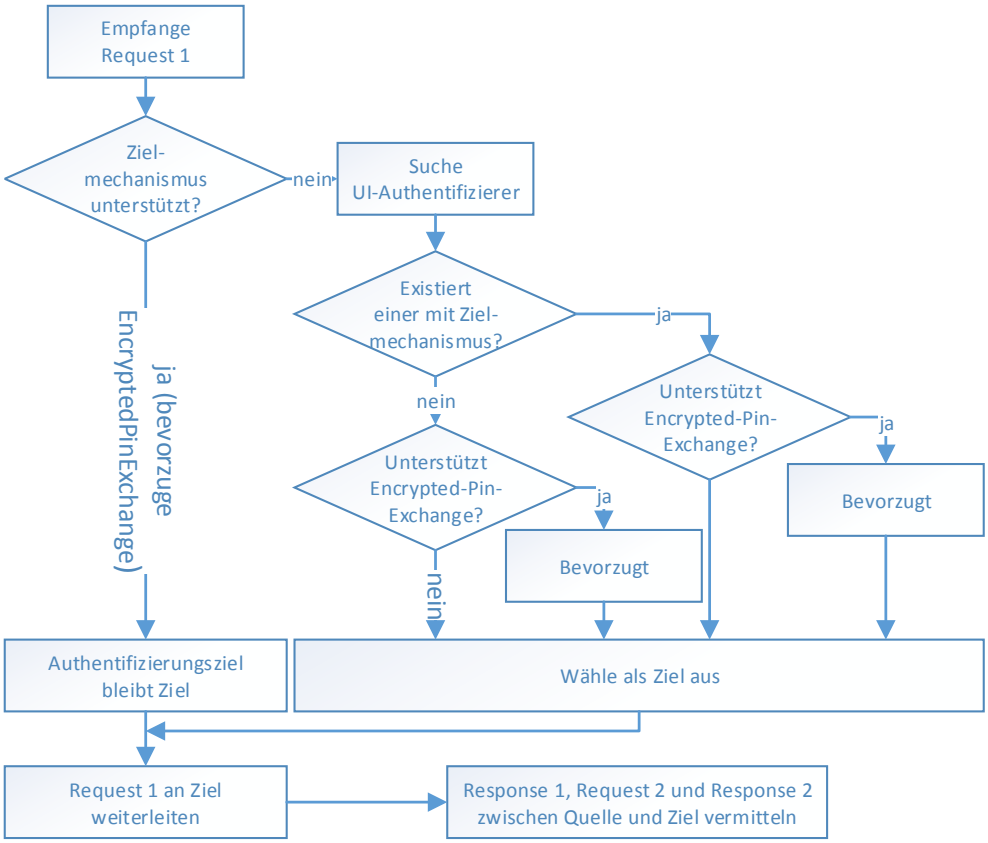
Das beschriebene Konzept spielt seine Stärke dann aus, wenn viele Mechanismen unterstützt werden, vornehmlich in einem mobilen Gerät wie einem Smartphone. Konkret diskutiert werden in diesem Kapitel die Authentifizierung über Flackern von sichtbarem Licht, Tapping und numerische Pins. Darüber hinaus sind zahlreiche zusätzliche Mechanismen denkbar. So ließe sich ein geflackertes Muster auch als Token per Nahfeldkommunikation (NFC) wei-



(a) Vorgänge im Client



(b) Vorgänge im Gerät



(c) Vorgänge im UI-Authenticator

Abbildung 4.15: Vorgänge in den verschiedenen Gerätegruppen während einer indirekten Authentifizierung

tergeben oder ein gescannter QR-Code in ein Audiosignal umwandeln. Zudem kann jedes beliebige Multimediagerät in der Nähe genutzt werden, ohne dass es vorher Vertrauensbeziehungen zu einem der Geräte oder einer vertrauenswürdigen zentralen Instanz aufbauen muss.

Dem entwickelten kryptografischen Protokoll ist dabei eine Einschränkung inhärent. Es erfordert, dass immer genau ein Teilnehmer ein Einweg-Passwort darstellen kann (Ausgabe) und genau ein Teilnehmer das Einweg-Passwort entgegennimmt (Eingabe). Das darauf aufbauende Kommunikationsprotokoll schränkt darüber hinaus ein, dass das Device die Ausgabe und der Client die Eingabe benötigt.

Die Einschränkung durch das Kommunikationsprotokoll lässt sich leicht lösen. Dazu wartet das Device nach seiner ersten Antwort auf den Eingang einer Pin die der Client ausgibt, nachdem er die erste Antwort erhalten hat. Jedoch darf (und kann) nun nicht das Device seinen öffentlichen Schlüssel mit dem Einweg-Passwort verschlüsseln. Stattdessen muss der Client dies übernehmen. Die Erkennung, in welche Richtung eine Pin ausgetauscht werden soll, lässt sich durch eine erweiterte Bezeichnung der Authentifizierungsmechanismen angeben.

Die Einschränkung durch das kryptografische Protokoll, dass genau ein Gerät eine Eingabe- und genau ein Gerät eine Ausgabemöglichkeit benötigt, führt hingegen zu einem Sicherheitsgewinn. In [KSTU09] und [DTH06] wird das Problem des *Rushing User* beschrieben. Es bezeichnet das Phänomen, dass ein Nutzer Aufgaben wie das Pairing von Geräten möglichst einfach gestaltet, auch, wenn es eine Schwächung der Sicherheit bedeutet. Besitzen nun zwei zu pairende Geräte nur eine Eingabemöglichkeit, ist es dem Nutzer möglich, einfache Zahlenkombinationen (bzw. ihre Repräsentationen) zu wählen. MITM-Angriffe mit Einweg-Pins wie „0000“ oder „123456“ wären demnach häufig erfolgversprechend [KED+12]. Ebenso stellt es sich dar, wenn ein Nutzer bestätigen muss, dass zwei Einweg-Pins identisch sind, wenn beide Geräte von einer Ausgabe Gebrauch machen. Häufig wird der Schritt ohne die notwendige Sorgfalt übergangen, ein MITM könnte demnach eine beliebige Pin frei wählen. Die erwähnte Einschränkung unterbindet dieses Problem, da die Pin nicht vom Nutzer sondern von einem der Geräte gewählt wird und der Nutzer diese anschließend korrekt auf das andere Gerät übertragen muss.

Eingangs wurde in Abschnitt 4.1 die Anforderung definiert, dass Diebstahl oder Verlust eines Multimediageräts nicht die Sicherheit des Netzwerks gefährden darf. Da das beschriebene Protokoll nur die Einweg-Pins zwischenspeichert, kann das Gerät einem unrechtmäßigen Besitzer keine Informationen über die aufgebauten Vertrauensverhältnisse liefern, da die Pins nicht in die gemeinsamen Schlüssel eingehen. Bestehende Vertrauensbeziehungen sind demnach nicht beeinträchtigt. Nach dem Zeitpunkt des Verlusts oder Diebstahls aufgebaute Verbindungen sind jedoch anfällig, wenn das fragliche Gerät für das Verfahren `ws4d:EncryptedPinExchange` eingesetzt wurde. Die bestehende Vertrauensbeziehung zu anderen Multimediageräten kann missbraucht werden, da das gekaperte Gerät im Falle einer Kaskadierung in Besitz der Einweg-Pin gelangt. Ein MITM-Angriff wäre so erfolgreich.

Zur Abwehr des Angriffs sind verschiedene Möglichkeiten denkbar. Zunächst kann (und muss) die Vertrauensbeziehung zu einem verlorenen oder gestohlenen Gerät gelöst werden. Anschließend kann das fragliche Gerät nicht mehr in die Kaskadierung einbezogen werden. Das kommt jedoch nur in Frage, wenn der Verlust des Geräts bemerkt wird. Zur Erhöhung der Sicherheit ist es denkbar, Vertrauensbeziehungen ein Ablaufdatum zu geben, sodass die Authentifizierung regelmäßig wiederholt werden muss, was jedoch die Nutzerfreundlichkeit erheblich beeinträchtigt. Ein Kompromiss liegt in der Möglichkeit, die ausgehandelten Schlüssel auf dem Gerät mit einem Nutzerkennwort oder z.B. biometrischen Merkmalen zu verschlüsseln. Soll der Schlüssel für einen Authentifizierungsvorgang genutzt werden, muss er zuvor entsperrt werden. Zwar ist hierfür das gestohlene Gerät selbst zuständig, und kann nicht etwa von einem der übrigen Netzteilnehmer übernommen werden, dafür ist die Sicherheit stark erhöht bei einer vergleichsweise niedrigen Auswirkung auf die Benutzbarkeit.

Abschließend soll die Entstehung des Pfades bei einer Kaskadierung diskutiert werden. Da Client und Multimediageräte während einer Authentifizierung keine Kenntnis über die Gesamtheit der zur Verfügung stehenden Geräte und ihre jeweiligen Authentifizierungsmethoden besitzen, ist es möglich, dass nicht der kürzeste Pfad gewählt wird, sondern eine nichtminimale Anzahl an Multimediageräten einbezogen wird.

Andernfalls müsste jedes einbezogene Gerät eine Topologie des gesamten Netzwerks erstellen und den kürzesten Pfad bestimmen. Da es sich bei den

betrachteten Anwendungsfällen um sehr dynamische Netzwerke handelt, in denen insbesondere Multimediageräte mit hoher Frequenz hinzukommen oder das Netzwerk verlassen, müsste die Erstellung dieser Topologie bei jedem Authentifizierungsvorgang erfolgen. Das erhöht das Datenaufkommen im Netzwerk beträchtlich. Hinzu kommt, dass der Client diesen Aufwand gezielt an ein beliebiges Multimediagerät abgibt, wodurch bereits bei der Wahl des ersten Geräts ein unvorteilhafter Pfad eingeschlagen werden kann. Die in dieser Arbeit entwickelte Lösung kommt ohne das Wissen über das gesamte Netzwerk aus und erfordert den kleinstmöglichen Aufwand für den potentiell ressourcenschwachen Client.

Kapitel 5

Anforderungsanalyse

In diesem Kapitel werden die Anforderungen analysiert, die an eine Sicherheitsinfrastruktur für verteilte eingebettete Systeme gestellt werden müssen. Hierzu werden zunächst in Abschnitt 5.1 Angreifer- und Bedrohungsmodelle für das Internet der Dinge und intelligente Umgebungen und Anwendungen wie Smart Homes oder das Ambient Assisted Living analysiert. Daraus lässt sich ableiten, welche Maßnahmen ergriffen werden müssen, um die in Kapitel 2.2.1 definierten Schutzziele zu gewährleisten. Anschließend werden funktionale Anforderungen an die zu entwickelnde Lösung erarbeitet. Im Abschnitt 5.2 erfolgt dies anhand abstrakter Empfehlungen in der Literatur und in Abschnitt 5.3 anhand zweier konsolidierter Szenarien. Die Anforderungen werden in Abschnitt 5.4 zu einem Anforderungskatalog zusammengefasst, der als Prüfmittel für die zu entwickelnde Lösung dient.

5.1 Allgemeine Angreifermodelle & Sicherheitsbetrachtungen

Absolute Sicherheit ist ein Zustand, der sich nicht erreichen lässt. Es ist jedoch möglich, durch bestimmte Maßnahmen das Risiko zu mindern, dass ein Angreifer unberechtigt Daten erhält oder unbemerkt verändert oder hinzufügt. Angreifermodelle beschreiben qualitativ die Fähigkeiten angenommener Angreifer hinsichtlich z.B. ihrer Kenntnisse, ihrer Ressourcen oder ihres Budgets ([Eck08]). So lassen sich notwendige Maßnahmen ableiten, einer Angreiferkategorie zu begegnen.

Dolev und Yao definieren in [DY83] das nach ihnen benannte Angreifermodell, das heute als das passende für Internetkommunikation akzeptiert ist ([Mao04, SB09]). Laut dem Dolev-Yao-Modell ist ein Angreifer ein legitimer Teilnehmer des Netzwerks, das er angreift. Das bedeutet, dass er grundsätzlich lesenden und schreibenden Zugriff auf jede Nachricht hat, die durch dieses Netzwerk geleitet wird und dass er wie jeder andere Teilnehmer auch die Kommunikation mit jedem anderen Teilnehmer initiieren kann. Mao leitet zusätzlich in [Mao04] folgende Eigenschaften aus dem Dolev-Yao-Modell ab. Ein Angreifer kann sich zwischen zwei Teilnehmern befinden und somit als Man-in-the-Middle (MITM) fungieren. Durch seinen uneingeschränkten Kanalzugriff ist es ihm möglich, Nachrichten im Transit zu verändern, zu duplizieren, zu löschen oder beliebig hinzuzufügen. Auch eine Neuadressierung (z.B. IP-Spoofing) ist ihm möglich. Es handelt sich hier demnach um einen aktiven Angreifer. Er kann nicht nur passiv das Geschehen beobachten und belauschen, sondern aktiv in die Kommunikation eingreifen und sie nach Belieben verändern. Mao stellt jedoch klar, dass der Angreifer nicht allmächtig ist. Insbesondere spricht das Dolev-Yao-Modell ihm Fähigkeiten ab, wie z.B. das Erraten kryptografisch sicherer Zufallszahlen, das Errechnen von Klartexten aus Chiffren und umgekehrt ohne passendes Schlüsselmaterial oder die Berechnung eines privaten Schlüssels aus einem öffentlichen.

Shelby und Bormann nehmen in [SB09] für das Internet der Dinge ebenfalls das Dolev-Yao-Angreifermodell an und ergänzen es um einen wichtigen Punkt. Im Gegensatz zu klassischen Rechnernetzen besteht bei verteilten eingebetteten Systemen eine weit höhere Wahrscheinlichkeit, dass ein Angreifer physische Kontrolle über ein Gerät erlangt. Um zu vermeiden, dass ein Angreifer diese Kontrolle nutzt, um an auf dem Gerät gespeichertes Schlüsselmaterial zu gelangen, sind spezielle Hardware-Schutzmaßnahmen nötig, die laut den Autoren aus Kostengründen mit hoher Wahrscheinlichkeit nicht umgesetzt sind. Daher muss vermieden werden, dass der Verlust eines Geräts die Sicherheit des gesamten Netzwerks kompromittiert.

Die Autoren Heer, Garcia-Morchon et al. haben in [HGMH+11] Sicherheitsanforderungen an das Internet der Dinge formuliert und diese kontinuierlich in [GMKK+13] erweitert. Sie heben besonders hervor, dass die sichere Interoperabilität von Geräten verschiedener Hersteller während des Verbin-

dungsaufbaus und im laufenden Betrieb gewährleistet sein muss. Außerdem müssen „Reownership and Recommissioning“ berücksichtigt werden, das heißt die Tatsache, dass Geräte verkauft und gebraucht gekauft werden. Neben bereits besprochenen Angriffen wie Belauschen, MITM oder Diebstahl, werden vor allem weitere Angriffe auf die Systemsicherheit beschrieben, wie z.B. das unbemerkte Ersetzen von Firmware oder Produktpiraterie. Abschließend verlangen die Autoren auch eine protokollübergreifende Ende-zu-Ende-Sicherheit. Es sei anzumerken, dass die Autoren in ihren Schilderungen voraussetzen, dass die Geräte des Internets der Dinge regelmäßigen Wartungszyklen durch Experten durchlaufen.

Eine weitere umfassende Analyse von Angreifermodellen und Bedrohungen im Internet der Dinge ist in [RZL13, S.2271] zu finden. Neben den bereits diskutierten Aspekten stellen die Autoren einen Vergleich zwischen zentralisierten und dezentralisierten Topologien an. Sie halten fest, dass Zentralisierung zu einem primären Angriffsziel führt, da sich Angriffe auf lohnenswerte Ziele konzentrieren. Es wird ein *Single Point of Failure* eingeführt, der zwar grundsätzlich stärkere Schutzmaßnahmen ergreifen kann, als die einzelnen Komponenten eines verteilten eingebetteten Systems, bei dem ein erfolgreicher Angriff jedoch auch fataler ist. Bei einem dezentralisierten System hingegen ist der Angriffsvektor weit größer, da es eine große Zahl an Angriffszielen gibt. Dafür ist die Menge an gestohlener Information bzw. beeinträchtigter Funktionalität bei einem erfolgreichen Angriff weit kleiner. Im Speziellen gehen die Autoren hier noch auf die starke Nutzerzentrik ein. Bei der Nutzung dezentralisierter Lösungen birgt eine mangelnde Bedienbarkeit das Risiko, versehentlich Sicherheitslücken zu öffnen, während bei der Wartung einer zentralisierten Lösung durch einen Experten die Gefahr der Kompromittierung des gesamten Netzwerks besteht. Sie fassen zusammen, dass keines der Paradigmen klare Vorteile dem anderen gegenüber hat.

Abschließend sei auf das Sicherheitsmodell für Smart Homes von Pishva und Takeda in [PT08] verwiesen. Es zeigt sich, dass es sich nicht von den bisher für das Internet der Dinge betrachteten Modellen unterscheidet. Sie verweisen jedoch zusätzlich auf die Anfälligkeit von Daten gegen Veränderung nicht nur während des Transits, sondern auch bezogen auf ihre Lagerung.

5.2 Anforderungen aus der Literatur

Schon als Mark Weiser 1991 seine Vision des Ubiquitous Computing in [Wei91] beschrieb, machte er auf mögliche Auswirkungen auf die Privatsphäre seiner Nutzer und die Notwendigkeit, diese mit kryptografischen Mitteln zu schützen, aufmerksam. Auch für die verschiedenen Inkarnationen wie Smart Homes oder Smart Office Spaces werden in der Literatur verschiedene Anforderungen definiert. Diese werden seltener als konkrete Funktionalität und häufiger als grundlegende Eigenschaften beschrieben. So schlagen Kagal et al in [KFJ01] ein Autorisierungskonzept mit delegierbaren Rechten vor, während in [CAMN+02] erkannt wird, dass ein Trend besteht, dass Autorisierungen nicht mehr manuell gepflegt, sondern automatisch ermittelt werden.

In [CAMN+02] und [AMRCM03] isolieren die Autoren eine umfangreiche Liste von erforderlichen Eigenschaften. So fordern sie, dass ein Sicherheitssystem für intelligente Umgebungen transparent und unaufdringlich sein soll. Sie schlagen verschiedene Sicherheitsstufen und die Fähigkeit Kontextinformationen zu berücksichtigen vor. Neben der Notwendigkeit der Skalierbarkeit einer Lösung nennen die Autoren zwei weitere wesentliche Punkte. Ein Sicherheitssystem für intelligente Umgebungen wie ein Smart Home muss die notwendige Flexibilität aufweisen, um in Umgebungen mit wenigen Ressourcen gut und in Umgebungen mit vielen Ressourcen noch besser zu funktionieren. Außerdem erkennen sie, dass die stattfindenden Authentifizierungsvorgänge nicht mehr auf den Nutzer ausgerichtet sein dürfen, sondern die Geräte als Teilnehmer erkennen müssen.

Martínez et al formulieren in [MLH+08] einige ganz konkrete Anforderungen wie Ende-zu-Ende-Sicherheit, verschiedene Sicherheitsstufen und die Möglichkeit zu feinstufig konfigurierbaren Zugriffsrechten. Müller et al. definieren in [MKGC09], dass ein Sicherheitskonzept einfach und intuitiv nutzbar sein muss und Nutzer und Administratoren „halb-automatisch“ bei der Registrierung unterstützen soll. Da sie einen zertifikatsbasierten Ansatz vorschlagen, fordern sie, dass sichergestellt wird, dass Zertifikate nur legitimen Teilnehmern ausgestellt werden und dabei MITM-Angriffe unterbunden werden. Diese Zertifikate sollen zur Authentifizierung gegenüber Diensten genutzt werden können.

Schließlich berücksichtigen sie das Szenario, Gästen die Möglichkeit zu geben, am Netzwerk teilzunehmen.

5.3 Szenarien

Neben der Analyse von Angreifermodellen und abstrakt formulierten Forderungen aus der Literatur, wurde in dieser Arbeit eine Vielzahl an Szenarien für intelligente Umgebungen analysiert, insbesondere, um funktionale Eigenschaften bestimmen zu können, die ein umfassendes Sicherheitskonzept für intelligente Umgebungen aufweisen muss. Im folgenden werden die untersuchten Szenarien für Smart Homes und Smart Office Spaces in Form ihrer funktionalen Beschreibung jeweils zusammengefasst und anschließend aus ihren Beschreibungen die notwendigen Eigenschaften abgeleitet.

5.3.1 Smart Home / Ambient Assisted Living

Im diesem Abschnitt erfolgt die Konsolidierung verschiedener Szenarien aus dem Bereich des Smart Home bzw. Ambient Assisted Living.

Funktionale Beschreibung

Die Arbeitsgruppen „Schnittstellenintegration und Interoperabilität“ und „Kommunikation“ der BMBF/VDE Innovationspartnerschaft AAL haben im Rahmen ihrer Arbeit eine umfassende Szenariensammlung für das Ambient Assisted Living erstellt [Ebe11]. Aus insgesamt 37 zusammengetragenen Szenarien wurden neunzehn prototypische Assistenz- und sechs Infrastruktursysteme isoliert, die sich in vielen der Szenarien finden ließen. Während es sich bei einigen dieser Systeme um reine Software-Lösungen handelt (z.B. „Ernährungsmanager“ oder „Personalisierte Wiedergabe von Nachrichten“), stellen andere ganz konkrete Forderungen an die eingesetzten Geräte, ihre Funktionalität und ihre Sicherheitseigenschaften.

Zu diesen gehört zunächst die Ad-hoc-Vernetzung von Multimediasystemen und mobilen Endgeräten in die existierende Infrastruktur, damit Medien von verschiedenen Speicherorten auf jedem verfügbaren Endgerät wiedergegeben werden können. Dazu gehört auch ihre Einbindung in das „Kommunikations-

system“, um z.B. Telefonie über alle verfügbaren Geräte zu ermöglichen. Das entwickelt sein volles Potential in der Verbindung mit der „ambienten Audiosteuerung“, bei der Musik oder ein Telefonat dem Nutzer durch seine Wohnung folgt und durch die „ambiente Gehöranpassung“ die Lautstärke der eventuellen Hörschwäche des Nutzers anpasst. Weitere ambiente Assistenzsysteme steuern die Beleuchtung inklusive Sonnenblenden oder überprüfen das Verhalten des Nutzers, um von eventuellen Anomalien auf Notfälle schließen zu können. Neben weiteren einfachen Assistenzfunktionen wie der Vitalparameterüberwachung im „intelligenten Bett“ oder den „intelligenten Badmöbeln“, die sich der Körpergröße ihrer Nutzer anpassen, durchdringen andere Funktionen den Wohnraum ihres Nutzers vollständig. Beispiele sind der „Energy Management Server“, „biometrische Zugangskontrollen“ und das Telemonitoring-System zur Fernüberwachung des Gesundheitszustands der Bewohners durch Angehörige und medizinisches Personal. Die isolierten Infrastruktursysteme beschreiben die zugrundeliegenden Mechanismen zur Umsetzung der Assistenzsysteme. Zu ihnen zählen ein Alarm- und ein Meldesystem und je eine Schnittstelle für die Fernwartung durch einen Dienstleister und die Ansteuerung der Haustechnik durch die Bewohner. Daneben werden ein Lokalisierungs- und Identifizierungssystem und die Möglichkeit zur Sprachsteuerung genannt.

Abgeleitete Eigenschaften

Tabelle 5.1 ordnet ausgewählte Subsysteme aus den beschriebenen Szenarien hinsichtlich ihrer verfügbaren Nutzerschnittstellen, ihren Kommunikationsprotokollen und -medien sowie ihrer Stromversorgung und verfügbaren Ressourcen ein. Daraus wird das Ausmaß der Heterogenität bezüglich all der betrachteten Parameter der eingesetzten Geräte ersichtlich.

Aus der Einordnung lassen sich drei Geräteklassen ableiten. Die Gruppe der Kleinstgeräte ist insbesondere durch ihre starke Ressourcenlimitierung charakterisiert. Sehr häufig sind die Geräte batteriebetrieben und kommunizieren drahtlos über Low-Power-Funkprotokolle wie ZigBEE bzw. 6LoWPAN, da diese Eigenschaften ihr Deployment stark vereinfachen und hohe Mobilität gewährleisten. Sehr häufig beschränken sich ihre Schnittstellen auf ihre primären Bedienelemente, wie z.B. Taster, LEDs oder einfache LCDs.

	Schnittstellen, Bedienelemente	Protokoll / Medium	Stromversorgung / Ressourcen
Sensoren (z.B. Rohrbruch)	Taster, LEDs, ggf. LCD	ZigBEE, 6LoWPAN / drahtlos	v.A. batteriebetrieben / schwach
User Tracking	Abhängig von Technologie. Bei Kameras Taster, LED, LCD, CCD...	IP, DLNA / Festinstallation, eher drahtgebunden	eher permanente Stromversorgung / schwach bis mittel
Multimediageräte (Ambiente Audiosteuerung / Gehöranpassung)	LCD, Fernbedienungen, Audio (in & out)	DLNA o.ä., / drahtgebunden oder drahtlos	größtenteils permanente Stromversorgung / mittel bis stark
Glühleuchten (ambiente Lichtsteuerung)	Optisch	PLC, Digitalstrom / drahtgebunden	permanente Stromversorgung / schwach
Beleuchtung: Schalter	Taster, optisch	ZigBEE, 6LoWPAN, IP / drahtgebunden oder drahtlos	verschiedenes denkbar: batteriebetrieben, permanent oder Energy Harvesting / schwach
Biometrische Zugangskontrolle	Abhängig von biometrischem Merkmal, evtl. Bedienfeld	proprietär, seriell / drahtgebunden	Permanent / schwach bis mittel
Energy Management Server	NFC, Kabel, evtl. komplexes Bedienfeld und umfassende Eingabemöglichkeit	Protokolle aus dem Smart Metering (M-Bus, DMLS/ COSEM, SML), PLC, digitalStrom, Ethernet / drahtgebunden	permanent / mittel bis stark
Medizinische Komponenten	einfache Anzeigen und Eingabemöglichkeiten	IP, DPWS / drahtgebunden oder drahtlos	je nach Gerät permanent (Heimtrainer) oder Batteriebetrieben (Pillenbox) / schwach bis mittel
z.T. globale Kommunikation nach außen (Übermittlung medizinischer Daten, Notfallmeldung) und nach innen (Ansteuerung / Überwachung der Haustechnik)			

Tabelle 5.1: Klassifikation ausgewählter Subsysteme aus den Szenarien

Dem gegenüber stehen Geräte, die über eine permanente Stromversorgung verfügen und demnach auch keine kritische Ressourcenlimitierung innehaben. Während auch sie drahtlos kommunizieren können, sind sie häufig parallel oder ersatzweise drahtgebunden vernetzt, was ihren Datendurchsatz erhöht. Auch ihre Schnittstellen sind häufig auf ihre Bedienelemente begrenzt, wobei es sich um detailreiche Anzeigeelemente und umfassende Eingabegeräte wie (virtuelle) Tastaturen handeln kann.

Als dritte Gruppe lassen sich Multimediageräte identifizieren. Hinsichtlich ihrer Ressourcen sind sie als mittel bis stark einzustufen, sie können über verschiedene Protokolle drahtgebunden oder drahtlos kommunizieren. Ihre Besonderheit liegt darin, dass sie sich im Gegensatz zu den vorigen Gerätegruppen nicht im Hintergrund halten, sondern sie befinden sich in unmittelbarer Nähe zum Nutzer. Häufig werden sie durch ihre hohe Mobilität direkt am Nutzer getragen. Darüber hinaus verfügen sie über eine Vielzahl an Bedienelementen, die ihnen als Schnittstellen dienen.

Neben dem hohen Maß an Heterogenität hinsichtlich Ressourcen, Protokollen und Schnittstellen, zeigen die Szenarien, dass auch sichere externe Kommu-

nikation, insbesondere über das Internet, gewährleistet sein muss. Dies betrifft vor allem Anwendungsfälle wie die Vernetzung mit der Praxis des betreuenden Arztes zur Übermittlung von Gesundheits- und Trainingsparametern und Notruf- bzw. Meldefunktionen. Während bei AAL-Szenarien in der Regel von initialen Installationen und Fernwartung durch einen Dienstleister ausgegangen wird, richten sich Smart-Home-Szenarien darauf aus, dass neue Geräte durch den Nutzer selbst in Betrieb genommen werden können.

5.3.2 Bürogebäude mit Smart Labs

Dieser Abschnitt betrachtet ein aus der Literatur konsolidiertes Szenario für ein intelligentes Bürogebäude.

Funktionale Beschreibung

Auch in Labor- und Bürogebäuden spielen ambiente Assistenzsysteme eine wichtige Rolle. So lässt sich Nutzertracking für ein intelligentes Beleuchtungssystem nutzen oder um zu erkennen, wenn sich ein Nutzer zu lange in einer Gefahrenzone aufhält. Ebenso wie im Smart Home lassen sich ambiente Kommunikationsanlagen einrichten, die ein freies Telefonieren erlauben und einen Anrufbeantworter aktivieren, um eine eventuell stattfindende Besprechung nicht zu unterbrechen ([Rö10]). Neben einer auf den Büronutzer angepassten Beleuchtungssteuerung ([CPC+04]) kann ein ambientes System auch den Stresslevel des Nutzers erfassen und evtl. eingehende Anrufe umleiten oder eine Pause empfehlen ([Rö10]).

Intelligente Türschilder wie in [BKS+08] beschrieben, erlauben dynamische Arbeitsplätze zur Auslastungsoptimierung z.B. für Außendienstmitarbeiter, die nur wenige Stunden jede Woche im Büro verbringen. Diese Schilder können ebenfalls ein stattfindendes Telefonat oder eine Besprechung anzeigen. In diesen Fällen, oder bei Abwesenheit, kann das Türschild auch entsprechende Notizen anzeigen. Die Türschilder können auch als Wegweiser für ein Szenario aus [Rö10] genutzt werden, in dem Gäste durch Richtungsangaben zu ihrem Ziel navigiert werden. Durch Nutzerlokalisierung und -identifizierung können dem Gast gleichzeitig die notwendigen Berechtigungen zum Betreten bestimmter Bereiche eingerichtet werden.

Intelligente Besprechungsräume nutzen ihre ambienten Systeme ebenfalls zur Steuerung der Beleuchtung und zur Ortung und Identifizierung von Nutzern. So lassen sich für einen Vortrag die Lichtverhältnisse anpassen und Vortragsinhalte können automatisch dargestellt werden. Sensoren in den Stühlen können die Aufmerksamkeit des Auditoriums erfassen und an den Vortragenden melden, damit dieser seinen Vortragsstil anpassen kann ([Rö10]). Auch das Zuschalten entfernter Teilnehmer ist vorgesehen. Das gilt einerseits für Zuhörer, jedoch auch für Vortragende, deren Präsentationsmaterial zur Verfügung stehen muss ([Rö10]). Spracherkennung soll es ermöglichen, automatisch Mitschriften einer Besprechung anzulegen, die allen Teilnehmern zur Verfügung stehen bzw. einzelnen Teilnehmer vorgelegt werden, wenn sie die Besprechung verlassen mussten. Ein weiterer wichtiger Punkt ist die Verbindung von Besprechungsräumen, um z.B. Sitzungen spontan verbinden zu können und Daten zwischen beiden auszutauschen.

Abgeleitete Eigenschaften

Grundsätzlich entsprechen die Eigenschaften eines intelligenten Bürogebäudes denen eines intelligenten Heims. Ein offensichtlicher Unterschied liegt in seiner möglichen Größe. Während eine Wohnung oder ein Einfamilienhaus aus wenigen Räumen besteht, die alle der selben organisatorischen Einheit angehören, können große Bürogebäude hunderte von Räumen beherbergen. Diese gehören häufig verschiedenen administrativen Untergruppen an, wie z.B. einzelnen Abteilungen eines Unternehmens. Ein weiterer Unterschied liegt in der Dynamik. In Smart Homes ändern sich die verwendeten Geräte und die Identität der Benutzer selten, in Bürogebäuden kann dies durch Gäste, Außendienstmitarbeiter oder in Co-Working-Spaces sehr viel häufiger der Fall sein. Zwar ist grundsätzlich davon auszugehen, dass die Haustechnik eine initiale Installation vornimmt und die Geräte regelmäßig wartet (Batterietausch, Firmware-Updates, etc...), jedoch müssen einfache Konfigurationsänderungen auch durch die Nutzer möglich sein.

5.4 Zusammenfassung der Anforderungen

Die notwendigen Anforderungen an die Kommunikationssicherheit werden in Tabelle 5.2 zusammengefasst. Diese Tabelle wird im weiteren Verlauf dieser Arbeit zur Beurteilung der entwickelten Lösungen verwendet. Die Anforderungen lassen sich in drei Gruppen einteilen. Sie betreffen ganz allgemeine Sicherheitsanforderungen und solche, die die Eigenheiten intelligenter Umgebungen widerspiegeln. Schließlich lassen sich Anforderung formulieren, die vor allem auf die Interoperabilität verschiedener Technologien abzielen.

Durch die betrachteten Angreifermodelle wird ersichtlich, dass die in Kapitel 2.2 genannten Schutzziele durch kryptografische Maßnahmen erfüllt werden müssen. Der verwendete Kommunikationskanal muss dabei durchgängig und zu jeder Zeit als kompromittiert betrachtet werden, so auch zur Zeit des Verbindungsaufbaus. Letzteres ist von zentraler Bedeutung, da insbesondere in Smart-Home-Szenarien von einem kontinuierlichen Deployment ausgegangen werden muss, da Nutzer wiederholt neue Geräte in bestehende Infrastrukturen einbringen. Ebenfalls wird häufig betont, dass einzelne Geräte leicht unter physische Kontrolle eines Angreifers fallen können, sei es durch Diebstahl, Verkauf oder Verlust. Es bedarf eines Mechanismus zum koordinierten Austragen eines Geräts aus einem Netzwerk. Sollte ein Diebstahl oder Verlust unbemerkt bleiben, muss sichergestellt werden, dass dieser Fall eine möglichst geringe Auswirkung auf das gesamte Netzwerk hat und die Notwendigkeit, die gesamte Einrichtung zu wiederholen, muss vermieden werden.

Weder in den Bedrohungsanalysen noch in der Literatur ist eine klare Empfehlung hinsichtlich der (De)zentralisierung der Systeme zu erkennen. Als Anforderung lässt sich jedoch formulieren, die Vorteile aus beiden Paradigmen zu nutzen und möglichst wenige Nachteile umzusetzen. Das deckt sich mit der Forderung, dass eingebettete Systeme eine zentrale Instanz nutzen können, jedoch auch in ihrer Abwesenheit funktionieren sollen. So sind ein Höchstmaß an Robustheit ohne Single Point of Failure sowie die Entlastung von Kleinstgeräten gewährleistet. Ebenfalls mehrfach gefordert ist die sichere Interoperabilität sowohl zwischen verschiedenen Protokollen als auch zwischen verschiedenen Herstellern. Das schließt eine übergreifende Ende-zu-Ende-Verschlüsselung mit ein. Vereinzelt findet sich die Forderung nach „transparenten und unauf-

Grundlegendes	Berücksichtigung des Dolev-Yao-Angreifermodells
	Vermeidung von Single Points of Failure
	Sicheres, kontinuierliches Deployment
	Sichere externe Kommunikation berücksichtigt
	Flexible Unterstützung unterschiedlicher Autorisierungskonzepte
	Organisation versch. administrativer Gruppen
	Verschiedene Sicherheitsstufen
	Skalierbar
	Einfache Benutzbarkeit ohne Einschränkung der Sicherheit
	Unterstützung sicherer Datenhaltung
Intelligente Umgebungen	Berücksichtigung der Leistungsheterogenität
	Entlastung von Kleinstgeräten
	Berücksichtigung der Heterogenität der Bedienelemente
	Geräte im Fokus statt Nutzer
	Schutz des übrigen Netzwerks bei Verlust eines Geräts
	Koordiniertes Abmelden eines Geräts
	Einrichtung / Wartung durch Experten und Nutzer berücksichtigt
	Kleine Konfigurationsänderung durch Nutzer
	Delegation von Rechten
Interoper.	Weit verbreitete / akzeptierte Basistechnologie
	Sichere Protokollinteroperabilität
	Sichere Herstellerinteroperabilität
	Interoperable Ende-zu-Ende-Sicherheit

Tabelle 5.2: Zusammenfassung der Anforderungen

dringlichen“ Sicherheitsmechanismen, häufiger lautet der Anspruch „möglichst nutzbar“ oder „einfach zu bedienen“. Je nach Einsatzgebiet wird eine zentrale Wartung durch einen Experten vorausgesetzt oder nicht. Als Anforderung lässt sich ableiten, dass beide Fälle berücksichtigt werden müssen.

Schließlich findet sich eine Vielzahl an möglichen Autorisierungskonzepten. Dies reicht von zentral gespeicherten, statischen Tabellen über die Berücksichtigung temporärer Befugnisse für Gäste hin zu kontextsensitiven Entscheidungssystemen. Eine zu entwerfende Infrastruktur muss demnach die Flexibi-

lität aufweisen, verschiedene Autorisierungskonzepte zu unterstützen. Sowohl Autorisierung als auch Authentifizierung dürfen nicht mehr klassisch auf die Benutzer ausgerichtet sein, sondern müssen die Geräte, deren Dienste und Clients selbst in den Fokus rücken. Das zu entwerfende System muss skalierbar sein und sollte verschiedene Sicherheitsstufen berücksichtigen.

Aus den betrachteten Szenarien wird die Heterogenität der eingesetzten Geräte deutlich, der eine zu entwickelnde Sicherheitsinfrastruktur Rechnung tragen muss. Das bedeutet zunächst, dass die unterschiedlichen Rechenleistungen der einzelnen Komponenten berücksichtigt werden und schwache Teilnehmer entlastet werden müssen. Es bedeutet ebenfalls, dass eine Strategie für die verschiedenen, teilweise stark eingeschränkten und zueinander inkompatiblen Bedienelemente gefunden werden muss. Die funktionale Beschreibung zeigt, dass Mechanismen für sichere eingehende und ausgehende Kommunikation berücksichtigt werden müssen. Um den Schutz der teilweise sehr sensiblen Daten auch über die Phase des Transits hinweg zu vereinfachen, sollten die Daten so versendet werden, dass eine verschlüsselte Speicherung vereinfacht wird.

Die Szenarien widersprechen sich hinsichtlich der Wartungsstrategie. Demnach muss als Anforderung formuliert werden, dass Installation und Wartung durch einen Experten mit eventueller anschließender Fernwartung sowie durch einen technischen Laien möglich sein muss. Gerade im Unternehmenseinsatz können größere Wartungsaufgaben durch Techniker vorgenommen werden, kleine Rekonfigurationen oder Änderungen wie z.B. die Nutzung von Geräten eines Raumes in einem anderen zur optimalen Ressourcenauslastung oder die Delegation von Zugriffsrechten müssen jedoch durch die Nutzer erledigt werden können.

Hinsichtlich des Szenarios des intelligenten Bürogebäudes muss eine Sicherheitsarchitektur hinreichend weit skalieren können, um potentiell Dutzende bis wenige hundert organisatorische Gruppen sicher miteinander zu verbinden. Schließlich ist es wünschenswert, sich einer technologischen Basis zu bedienen, die im Unternehmensumfeld bereits verbreitet ist. Durch eine bestehende Expertise ist eine hohe Akzeptanz der zu entwickelnden Lösung zu erwarten.

Kapitel 6

DPWSec: Das Devices Profile for Web Services Security

Einige zugrundeliegende Ansätze wie Client-Identifizierung oder die Unterscheidung in unsicheres und sicheres Discovery und Description wurden bereits in der folgenden Publikation im Rahmen der Anfertigung dieser Dissertation formuliert:

> *Sebastian Unger, Elmar Zeeb, Holger Grandy, Frank Golatowski & Dirk Timmermann: „Extending the Devices Profile for Web Services for Secure Mobile Device Communication“ in 4th International Workshop on Trustworthy Internet of People, Things & Services colocated at the Internet of Things 2010, ISBN: 978-1-4244-7414-1, Tokyo, Japan, November 2010 [UZG+10]*

Das Profil DPWSec, die Methodik zu seiner Erarbeitung und die sich daraus ergebenden Konzepte wurden bereits im Rahmen der Anfertigung dieser Dissertation wie folgt publiziert:

> *Sebastian Unger & Dirk Timmermann: „DPWSec: Devices Profile for Web Services Security“ in 2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing: EWSN-AP 2015, Singapur, Singapur, April 2015 [UT15]*

In diesem Kapitel wird das Devices Profile for Web Services Security vorgestellt. Es handelt sich dabei um ein Sicherheitsprofil für DPWS, das auf der Web Services Security Spezifikationssuite beruht und wurde in Anlehnung daran DPWSec genannt. DPWSec bedient sich der zuvor beschriebenen Arbeiten.

So basiert die Sicherheit auf Nachrichtenebene auf dem in Kapitel 3 beschriebenen Konzepten des WS Compact Security, und die in Kapitel 4 vorgestellte indirekte Authentifizierung wird vollständig und nahtlos integriert. Die in Kapitel 5 erarbeitete Anforderungsliste dient der abschließenden Bewertung.

Im Folgenden wird der Stand der Technik hinsichtlich Sicherheit für DPWS und verwandte Sicherheitskonzepte auf Anwendungsebene beschrieben und diskutiert. Die Web Service Security Spezifikationssuite stellt eine weit verbreitete und langzeiterprobte Sicherheitslösung für verteilte Systeme dar und ihre zugrundeliegende Basistechnologie – Web Services – wurde als DPWS bereits auf eingebettete Systeme adaptiert. Daher wird im weiteren Verlauf dieses Kapitels eine Methodik entwickelt, anhand derer die Kernkonzepte der bereits existierenden Web Service Security Spezifikationssuite isoliert und für die Verwendung durch verteilte eingebettete Systeme bewertet werden sollen. Dadurch soll eine Untermenge der ursprünglichen Spezifikationen, das Devices Profile for Web Services Security, erarbeitet werden. Die entstehenden Konzepte werden vorgestellt und ihre prototypische Implementierung beschrieben.

6.1 Stand der Technik

Dieser Abschnitt diskutiert zunächst das Sicherheitskonzept von DPWS, das auf austauschbaren Profilen besteht und es werden entwickelte Profile besprochen, die nicht Teil der Basisspezifikation sind. Darüber hinaus werden die Sicherheitsmechanismen verwandter Technologien untersucht und bewertet.

6.1.1 Sicherheit in DPWS

Wie es in der Welt der Web Services üblich ist, basiert auch das Sicherheitskonzept von DPWS auf sogenannten Profilen ([OAS09a, Kapitel 6]). In diesem Zusammenhang kann ein Profil als Sammlung von Vereinbarungen und Regelungen verstanden werden, auf die sich zwei oder mehr Kommunikationsteilnehmer vor der ersten Kommunikation einigen. Die Verwendung solcher Profile stellt einen ausgesprochen flexiblen Mechanismus dar, da zu jedem Anwendungsfall ein passendes Profil entwickelt werden kann, ohne die Spezifikation zu verletzen. In Ergänzung zur Definition des eigentlichen Mechanismus defi-

niert die DPWS-Spezifikation ein solches Sicherheitsprofil. Dieses Profil fußt auf den drei Begriffen „kompakte Signatur“, „sicherer Kanal“ und „Zertifikat“.

Als *kompakte Signatur* kommen die WSDD Compact Signatures zum Einsatz, die in WS Discovery beschrieben werden ([OAS09b, Kapitel 8.2]). Sie stellen eine kompakte Darstellungsform klassischer XML-Signature-Formate dar und wurden ausführlich in Kapitel 3.2.1 beschrieben. Sie werden eingesetzt, um Integrität und Authentizität von Dicovery-Nachrichten zu gewährleisten, die über das verbindungslose User Datagram Protocol (UDP) versendet werden. Als *sicherer Kanal* wird eine mittels TLS abgesicherte Verbindung zwischen zwei Knoten bezeichnet, die, wie in den Kapiteln 2.2.3 und 3.1.1 beschrieben, X.509.v3-*Zertifikate* zur Authentifizierung benötigen.

Ein Device zeigt die Unterstützung des Profils an, indem seine Transportadresse mit `https` statt `http` beginnt. Dadurch wird zunächst deutlich, dass ein Client seine Discovery-Nachrichten durch kompakte Signaturen schützen sollte und gleiches vom Device erwarten kann. Strenger ist das Profil beim Abruf von Metadaten. Da dies über TCP geschieht, muss diese Kommunikation durch TLS gesichert sein. Hosted Services können sich gesichert oder ungesichert ansprechen lassen. Sie zeigen dies an, in dem ihre Transportadresse mit `https` respektive `http` beginnt.

Ein Device muss sich immer durch Bereitstellung seines Zertifikats authentifizieren. Für Clients besteht alternativ die Möglichkeit, sich per HTTP-Authentifizierung – also unter Angabe von Nutzername und Passwort – auszuweisen. Die Spezifikation regelt jedoch nicht, wie Zertifikate oder Nutzername-Passwort-Kombinationen ausgetauscht werden müssen.

6.1.2 Weitere Sicherheitsprofile für DPWS

Das offene und flexible Konzept der Sicherheitsprofile für DPWS hat zu einer Reihe von Profilvorschlägen geführt, die hier in der Reihenfolge ihrer Erscheinung aufgeführt werden.

Martinez et al. schlagen in [MLH+08] ein Sicherheitsprofil für DPWS vor, das in Office-Spaces bzw. in industriellem Umfeld Anwendung finden soll. Die Autoren sprechen sich dabei für die Verwendung von WS Security und WS Trust aus (vgl. Kapitel 2.4.1 und 2.4.3). WS Security soll hierbei Vertraulichkeit, Integrität und Authentizität auf Nachrichtenebene gewährleisten, was

mit Ende-zu-Ende-Verschlüsselung und der Unabhängigkeit von bestimmten Protokollen in niederen Kommunikationsschichten motiviert wird.

Es werden drei Sicherheitsszenarien dargestellt, in denen Authentifizierung und Autorisierung lokal geregelt werden oder externe Instanzen entweder nur Authentifizierung oder beides übernehmen. Obwohl nicht explizit erwähnt, erinnern die Szenarien an die Einsatzfälle von WS Trust. Um diese mittels DPWS zu gewährleisten, werden nicht die klassischen DPWS-Rollen von Client und Device genutzt, sondern sog. SODA-Clients und SODA-Devices, die je einen Client- und einen Server-Teil beinhalten.

Das vorgeschlagene Profil ist durch die Nutzung der WS Security Spezifikationssuite für eine hohe Anzahl von Teilnehmern ausgelegt, jedoch wurde nicht berücksichtigt, dass ein solches Netzwerk auch viele Teilnehmer mit starker Ressourcenbeschränkung beinhaltet. Als Beispiel für die Authentifizierung werden X.509-Zertifikate angeführt, es werden jedoch auch beliebige andere Möglichkeiten offen gelassen. Angaben zur Verteilung der Credentials fehlen. Fallen die eingeführten Authentifizierungs- bzw. Autorisierungsserver aus, ist keine sichere Kommunikation mehr möglich.

Einen im Vergleich dazu minimalistischen Ansatz schlagen Hernández et al. in [HLP+09] vor. Auch sie sprechen sich für den Einsatz der WS Security-Basisspezifikation aus, und schlagen zunächst vor, zwei netzwerkweite symmetrische Schlüssel K_1 und K_2 zu nutzen, die OOB ausgetauscht werden müssen. Hierzu wird z.B. eine USB-Kabelverbindung vorgeschlagen. Die laufende Kommunikation soll mit K_1 verschlüsselt werden, so auch regelmäßige Kontrollnachrichten, die einen neuen Schlüssel beinhalten. Es wird vorausgesetzt, dass eine Security Management Console den Zustand aller Geräte überwacht, damit kein Teilnehmer die Verteilung neuer Schlüssel verpasst. Wird einer der Schlüssel als kompromittiert betrachtet, wird der jeweils andere genutzt, um ihn auszutauschen.

Darüber hinaus schlagen die Autoren vor, den Body einer SOAP-Nachricht immer zu verschlüsseln. Das soll "on the fly" geschehen, um den Speicher der Geräte möglichst effizient zu schonen. Um beim Signieren von Nachrichten ähnliche Vorteile zu erzielen, wird eingeführt, dass mehrere Nachrichtenteile, mindestens jedoch Header und Body, getrennt signiert werden. Das soll die im Speicher vorzuhaltenden Nachrichtenmengen minimieren.

Das in [HLP+09] vorgeschlagene Profil soll die Bedürfnisse von Geräten mit "10KB of run-time memory" berücksichtigen. Die vorgeschlagenen Netzwerkschlüssel bergen die reale Gefahr, dass bei Verlust eines Geräts die gesamte Netzwerksicherheit kompromittiert ist, und das Security-Bootstrapping vollständig wiederholt werden muss. Die vorgeschlagene Trennung von Header- und Body-Signaturen ermöglicht ein reales Angriffsszenario. Eine subtilere Eigenschaft kryptografischer Signaturen ist, dass sie Daten aneinander bindet. Werden Informationen gemeinsam signiert, sind sie untrennbar miteinander verknüpft. Die unabhängigen Signaturen hebeln diese Eigenschaft aus. Ein MITM-Angreifer kann frische Header mit früher aufgezeichneten Body-Teilen verknüpfen. Replay-Angriffe sind dadurch möglich.

In [UZG+10] wird ein Sicherheitsprofil für DPWS vorgeschlagen, dessen Einsatzzweck in der Vernetzung von wenigen Komponenten eines Infotainment-Systems im Automobilbereich liegt. Es setzt ebenfalls eine Verschlüsselung auf Nachrichtenebene ein und basiert auf der Verwendung selbstsignierter X.509-Zertifikate, die ausgetauscht und über ein OOB-Challenge-Response-Protokoll auf Basis numerischer Pins authentifiziert werden.

Obwohl es für automobile Anwendungsfälle entwickelt wurde und für wenige, vergleichsweise starke Teilnehmer ausgelegt ist, beinhaltet es Konzepte, die auch in dem in diesem Kapitel vorgestellten Profil DPWSec Anwendung finden, wie z.B. die Unterscheidung von sicherem und unsicherem Discovery, Client-Identifizierung und Beschreibung der Anforderungen einzelner Dienste oder Operationen in der Dienstbeschreibung. Die Konzepte werden später im Rahmen von DPWSec diskutiert.

Ein weiteres Sicherheitskonzept für DPWS schlagen Müller et al. in [MKGC09] und [MKGC10] vor. Sie definieren zunächst in [MKGC09] eine Netzwerkarchitektur, die vorsieht, dass ein Heimnetzwerk in einen sicheren und einen unsicheren Bereich unterteilt wird. Ein neues Gerät betritt zunächst den unsicheren Bereich, um ein eigenes Zertifikat von einer netzwerkeigenen CA signieren zu lassen. Um MITM-Angriffe zu verhindern, muss das neue Gerät Angaben wie Name und E-Mail-Adresse des Nutzers bereitstellen, die einem Administrator zum Zwecke der Entscheidungsfindung vorgelegt werden. Daraufhin wird ein Pin OOB übertragen, mit der das Protokoll authentifiziert werden soll. Hierzu wird die Pin mit dem privaten Schlüssel des neuen Ge-

räts verschlüsselt und an die CA übertragen, die die Pin entschlüsselt, ihre Richtigkeit überprüft und schließlich das Zertifikat des neuen Geräts signiert. Abschließend kann das neue Gerät den sicheren Bereich des WLANs mithilfe des von der CA signierten Zertifikats betreten. Die Zutrittskontrolle erfolgt hier mittels des Protokolls RADIUS ([RWRS00, IEE04]), das eine Zugangskontrolle durch Zertifikate ermöglicht.

Bei dem beschriebenen Protokoll zur Verteilung der Zertifikate ist den Autoren ein Fehler unterlaufen. Einem MITM-Angreifer ist es möglich, sich für das neue Gerät auszugeben und sich an seiner statt zu authentifizieren. Hierzu tauscht der Angreifer das zu signierende Zertifikat durch sein eigenes aus, wobei er die Angaben zu z.B. Nutzername und E-Mail-Adresse übernimmt. Da das neue Gerät die OOB ausgetauschte Pin mit seinem eigenen privaten Schlüssel verschlüsselt, ist es dem Angreifer möglich, sie mit dem öffentlichen Schlüssel des neuen Geräts zu entschlüsseln und kommt so in ihren Besitz, verschlüsselt sie neu mit dem eigenen privaten Schlüssel und lässt sich anschließend von der CA authentifizieren. Zwar ließe sich diese Schwachstelle beheben, in dem ein Challenge-Response-Protokoll durchgeführt wird, wie es in Kapitel 2.2.2, bzw. in [UZG+10] beschrieben wird. Jedoch bleiben verschiedene Nachteile bestehen. So richtet sich das Konzept auf Benutzer und nicht auf Geräte aus. Das verlangt von jedem Teilnehmer nicht nur die Verwendung von Zertifikaten, sondern auch die Bereitstellung von Nutzernamen oder E-Mail-Adressen. Schließlich wird RADIUS als zugrundeliegende Technologie vorausgesetzt ebenso wie WLAN oder Ethernet als Kommunikationsmedium. Dabei handelt es sich um Technologien, die nicht für den Einsatz auf verteilten eingebetteten Systemen vorgesehen sind.

Anschließend definieren Müller et al. in [MKGC10] eine Architektur zur Zugriffskontrolle für Dienste auf DPWS-Geräten. Hierzu bedienen sie sich der eXtensible Access Control Markup Language (XACML, [Mos05]). XACML stellt ein Nachrichten-Framework bereit, um im Moment einer Anfrage dynamisch eine Autorisierung einzuholen. Hierzu muss ein angesprochener Dienst als Client fungieren, um eine Autorisierungsentscheidung von einer definierten, vertrauenswürdigen Instanz, dem Policy Decision Point (PDP), einzuholen. Der PDP selbst trifft seine Entscheidungen auf Basis statischer Policies, sowie Attributen, die ihm über den Anfragesteller zur Verfügung stehen.

6.1.3 Sicherheit in weiteren Standards zur Gerätekommunikation

Neben DPWS existieren weitere standardisierte Technologien zur spontanen Vernetzung ressourcenbeschränkter Geräte. Diese werden an dieser Stelle zusammen mit ihren Sicherheitskonzepten vorgestellt.

UPnP & DLNA

Universal Plug and Play (UPnP, [UPn08, Mic00]) ist eine weit verbreitete Technologie zur spontanen Vernetzung von Geräten. Sie gilt als Vorläufer von DPWS und ist in vielen handelsüblichen Endkundenprodukten wie Druckern, Scannern und Home-Entertainment-Komponenten vorhanden. Für letztere Gerätegruppe existiert darauf aufbauend die Spezifikation der Digital Living Network Alliance (DLNA, [Dig12, All06]). Sie definiert eine Architektur für Multimediaanwendungen im Heimbereich und ermöglicht z.B. das Streaming von Audio- und Videodateien zwischen verschiedenen Geräten, ohne dass der Nutzer zuvor die Vernetzung konfigurieren muss.

Die Kernspezifikation von UPnP enthält selbst keine Sicherheitskonzepte, es existieren jedoch offizielle Ergänzungen [Ell03a, Ell03b], die ausführlich beschrieben werden [Ell03c]. In [Sel06] und [Hem06] kommen die Autoren jedoch zu dem Ergebnis, dass diese optionalen Sicherheitsmechanismen in der Praxis nie umgesetzt werden. Auch DLNA definiert keine Mechanismen zu den grundlegenden Schutzzielen. Lediglich die Umsetzung von DRM-Richtlinien zum Schutz von Urheberrechten ist geregelt.

Bluetooth

Der Standard IEEE 802.15.1, der gemeinhin unter dem Namen Bluetooth firmiert ([Blu07]), definiert eine weit verbreitete Technologie, die den Endkundenmarkt heute vollständig durchdringt. Sie kommt überall dort zum Einsatz, wo sich wenige (bis zu acht) Geräte spontan ohne existierende Infrastruktur vernetzen wollen und vergleichsweise geringe Bandbreiten und Entfernungen ausreichen. Größenordnungen liegen hier bei 1 MBit/s und 10 Metern. Beispiele sind Vernetzung von (tragbaren) PCs oder Smartphones ohne WLAN, Anbin-

dung von Peripherie wie Druckern, Lautsprecherboxen oder Headsets, Audio-Streaming, spontaner Dateiaustausch oder drahtlose Sensornetze ([KKP14]).

Das Sicherheitskonzept von Bluetooth kennt hier zunächst die Begriffe *Pairing* und *Bond*. Das Pairing ist ein Vorgang, in dem zwei Geräte nach OOB-Austausch einer Pin auf sichere Weise einen gemeinsamen Schlüssel aushandeln. Der Nutzer muss dazu meist die Pin festlegen und OOB übertragen. Hat eines der Geräte keine Möglichkeit zum Austausch einer numerischen Pin, so wird während der Produktion ein unveränderlicher Wert festgelegt. Das Ergebnis des Pairing ist ein Bond, die Geräte haben also eine gemeinsame Bindung und können fortan ohne weitere Nutzerinteraktion sicher miteinander kommunizieren ([KKP14]). Mithilfe des aufgebauten Schlüsselmaterials sind Verschlüsselung, Integritäts- und Authentizitätsprüfung möglich. Für Punkt-zu-Punkt-Verbindungen werden eigens abgeleitete Schlüssel genutzt, für sichere Gruppenkombination der eigentliche Master-Key ([SP08]).

Ende das Jahres 2009 wurde Bluetooth in Version 4.0 verabschiedet [Blu10a]. Dieser ist nicht abwärtskompatibel zu den Vorgängerversionen, beinhaltet jedoch den „Low-Energy-Stack“. Bluetooth Low Energy (BLE[1]) ist eine vollständig unabhängige Technologie, die auf „ultra low power“ - Anwendungen optimiert ist, und mit der es möglich sein soll, Geräte wie eine Pulsuhr ein Jahr lang mit einer Knopfzelle zu betreiben [Blu10b]. Das Sicherheitskonzept unterscheidet sich zu seinen Vorgängerversionen lediglich in Details wie dem verwendeten Algorithmus.

ZigBee

Bei ZigBee handelt es sich um eine weitere Technologie zur drahtlosen Vernetzung stark ressourcenbeschränkter Geräte und sie setzt auf der Funkübertragungstechnologie IEEE 802.15.4 ([IEE06]) auf. ZigBee definiert ebenso wie Bluetooth einen vollständigen Protokollstapel von der Bitübertragungsschicht bis zur Anwendung, der jedoch eine geringere Komplexität aufweist, einen geringeren Protokoll-Overhead verursacht und stromsparender arbeiten kann. Damit einher geht eine geringere maximale Übertragungsrate von 250 Kb/s

[1] Häufig werden auch andere Abkürzungen verwendet wie z.B. BTLE, B4LE, BT4LE, etc. . .

([LSS07, KS07]). Jedoch unterstützt ZigBee bis zu 65.000 Geräte in einem Netzwerk ([MSSS09]).

Kommunikationssicherheit ist integraler Bestandteil des ZigBee-Protokollstapels. So können die Daten bereits auf der Sicherungsschicht mit dem AES-Algorithmus im CBC-Modus geschützt werden. Die verwendete Schlüssellänge beträgt 128 Bit. Auf den höheren Schichten wird geregelt, welche Sicherheitsstufe und welche Schlüssel verwendet werden. Die Einstellungen gelten dabei für ein gesamtes Netzwerk. Die Sicherheitsstufe ist ein Wert zwischen 0 und 7 und gibt an, ob verschlüsselt wird (4-7) oder nicht (0-3) und wie lang der Message Integrity Code (MIC) zur Integritätsprüfung ist (0, 32, 64 oder 128 Bit). Der verwendete symmetrische Schlüssel wird immer netzwerkweit eingesetzt und wird entweder zur Produktionszeit auf dem Gerät gespeichert oder zur Laufzeit vor seiner Verwendung unverschlüsselt in-band auf das Gerät übertragen. Ein Belauschen dieser Phase kompromittiert demnach das gesamte Netzwerk. Der „Commercial Security Mode“ definiert zwar einen Master-Key, aus dem der Netzwerkschlüssel sowie einzelne Knotenschlüssel abgeleitet werden können, jedoch unterliegt die Übertragung des Master-Keys den selben Einschränkungen ([KS07]).

6LoWPAN

„IPv**6** over **lo**w-power **w**ireless **p**ersonal **a**rea **n**etworks“ (6LoWPAN, [KMS07, MKHC07]) ist das Ergebnis der Bemühungen, den Trend des „IP-over-Everything“ bis hin zu den Komponenten eines WPAN zu führen. So können diese Komponenten direkt über das zukunftssichere IPv6 adressiert werden, ohne dass Gateways eine Übersetzung in alternative Technologien oder Adressierungsschemata vornehmen müssen. Wie bei ZigBee kommt auf den unteren Schichten IEEE 802.15.4 zum Einsatz, das eine stromsparende Funkkommunikation ermöglicht. Auf der Schicht darüber, also Layer 3 im ISO-OSI-Schichtenmodell, befindet sich IPv6. Eine Zwischenschicht übernimmt notwendige Anpassungen von IPv6 an die Besonderheiten von IEEE 802.15.4, wie z.B. die maximale Framelänge von 127 Byte.

Mit der Verwendung von IEEE 802.15.4 bietet 6LoWPAN auch die Verschlüsselung der Frames mit AES-CBC128. Diese ist wie alle Layer-2-Mechanismen jedoch nur innerhalb eines Subnetzes wirksam. Vor dem Hintergrund

der Anbindung der Geräte mittels IPv6 an das Internet reicht dies demnach nicht aus. 6LoWPAN selbst definiert keine weiterführenden Sicherheitsmechanismen. Shelby und Bormann beschreiben in [SB09], dass sich das ESP-Protokoll der IPsec-Spezifikationsgruppe nutzen ließe, um Authentizität, Integrität und Vertraulichkeit zu gewährleisten. Zum Austausch der Schlüssel machen sie jedoch keine Angaben, sondern weisen lediglich auf Fehler hin, die es zu vermeiden gilt, wie z.B. die Nutzung netzwerkweiter Schlüssel.

6.1.4 Sicherheit in verwandten Forschungsarbeiten

Neben den besprochenen standardisierten Technologien für Gerätekommunikation wurde auch ein umfassender Überblick über existierende Forschungsarbeiten zu derartigen Technologien erstellt, um ihre Sicherheitsmechanismen zu untersuchen. Im Folgenden werden die betrachteten Technologien und ihre Sicherheitsmechanismen in aller Kürze dargestellt. Eine Zusammenfassung erfolgt in der Diskussion in Abschnitt 6.1.5.

Microsoft HomeOS [DMA+10, DMA+12] ist ein ganzheitliches Konzept von Microsoft zum Aufbau von Smart Homes. Es bietet nicht nur eine Kommunikationsarchitektur sondern daneben auch ein Betriebssystem, das auf jedem Gerät ausgeführt werden muss ebenso wie ein Ökosystemkonzept, das mit einem App-Store vergleichbar ist. Ein zentraler Vermittler muss die Kommunikation der heterogenen Komponenten in Treibern kapseln. Das vorgeschlagene Sicherheitskonzept besteht im wesentlichen aus einem Autorisierungssystem, dass nicht nutzer- sondern anwendungszentrisch ist, und das zeitlichen Aspekten wie dem Auslaufen einer Berechtigung große Aufmerksamkeit widmet.

Bei Amigo [IST04, NMT+07, AGR+05, RJPS05] handelt es sich um eine SOA-basiert Kommunikationstechnologie, die auf Web Services basiert und ebenso wie DPWS auf Spezifikationen wie WS Addressing, WS Discovery und WS Eventing aufbaut, jedoch nicht die Eigenheiten eingebetteter Systeme berücksichtigt. Die Authentifizierung der Teilnehmer übernimmt das Kerberos-Protokoll auf Basis von Passwörtern. Anschließend schlagen die Autoren eine symmetrische Verschlüsselung vor, ohne weitere Details zu nennen. Weiterhin soll die Sicherheit nicht Teil der Middleware sein, sondern als Service allen Teilnehmern angeboten werden. Die Autoren sehen darin potentiell den Vorteil, dass sich die Kompatibilität zwischen Komponenten erhöht. Die Autoren ge-

stehen allerdings ein, dass dies mindestens bei der Verschlüsselung von Daten zu Problemen führen wird.

Hydra ([ERA09]) ist das Ergebnis des gleichnamigen EU-Forschungsprojekts und bietet eine umfangreiche Middleware auf Basis von Web Services an. Während es gezielt die Einbindung von Sensoren zur Aufgabe hat, baut es nicht auf existierenden Standards wie DPWS auf, sondern setzt eigens entwickelte Konzepte um. Die Entwicklung kam im Jahr 2010 zum Erliegen. Später wurde die Software aus namensrechtlichen Gründen in Linksmart umbenannt und ist unter dem neuen Namen verfügbar. In [ERA09] verweisen die Autoren lediglich auf die Sicherheitskonzepte von Web Services. Weitere Veröffentlichungen behandeln nicht die Wahrung der grundlegenden Schutzziele, sondern untersuchen höhere Sicherheitsfunktionen wie Policy-Modelle ([HBE+07]), Secure-Flow-Konzepte ([AB09]) oder Untersuchungen zur Nutzung genetischer Algorithmen ([ZSIH09]).

Mit SM4ALL setzt ein weiteres EU-Projekt auf Web Services bei der Kommunikation im Smart-Home-Bereich. Ziel des Projektes ist, eine Middleware bereitzustellen, die von „jung und alt“ bedient werden kann ([BDCM+09]). Trotzdem keine Forschungsarbeit in diesem Umfeld Sicherheit in SM4ALL behandelt, schlägt das Ergebnisdokument D6.2a ein Sicherheitskonzept vor ([BCQ+10]). Darin kommt zunächst WS Security zum Schutz einzelner Nachrichten zum Tragen. Bei der Authentifizierung wird darin unterschieden, ob sich Nutzer außerhalb des Hauses befinden oder innerhalb. Für ersteres wird die Verwendung von Kerberos und Passwörtern vorgeschlagen, für letzteres eine Identifikation mittels RFID-Tags. Es wird vorausgesetzt, dass hinreichend viele Tags auch für Gäste vorhanden sind.

Die Middleware ubiSOAP ([CRI10]) ist Teil des IST-Forschungsprojekts PLASTIC, das Entwicklungsumgebung, SOA Middleware und Validierungsframework für pervasive Umgebungen anstrebt, deren Fokus auf mobilen Endgeräten mit SIM-Karten liegt ([PLA08]). ubiSOAP basiert auf Web Services, macht jedoch keine Angaben zu Sicherheitskonzepten. PLASTIC hingegen beruft sich auf die Möglichkeiten der WS Security Spezifikationssuite und nutzt die Basisspezifikation ([BCC+07]).

PECES ([ZSS11]) ist eine Middleware zur Vernetzung eingebetteter Systeme, die die objektorientierte Kommunikationstechnologie BASE ([BSGR03])

nutzt. Die Sicherheitskonzepte basieren auf Zertifikatshierarchien zur Authentifizierung und optionalem, anschließenden Austausch eines symmetrischen Schlüssels. Zur Autorisierung wird ein Rollenmodell verwendet. Der Anwendungsentwickler ist für Zugriffsrichtlinien und Austausch von Zertifikaten zuständig ([HHA+10])

Ebenfalls objektorientiert arbeitet Gaia ([RHC+02]) auf Basis von CORBA. Gaias Sicherheitskonzept ist in [CAMN+02] beschrieben. Um mögliche Kompromisse zwischen der Stärke der Authentifizierung und der Unaufdringlichkeit zu finden, wird Vertrauen in einem Wert zwischen 0 und 1 ausgedrückt, wobei verschiedene Authentifizierungsmethoden zu verschiedenen Werten führen. Höhere Vertrauenswerte führen zu weitreichenderen Befugnissen. Zur eigentlichen Authentifizierung werden Kerberos und Passwörter genutzt. Die eigentliche Kommunikation übernimmt MIST ([AMCK+02]), die Vertraulichkeit und Privatsphäre verspricht.

Der Ableger Mobile Gaia ([CAMCM05]) ist auf die Verwendung auf mobilen Endgeräten spezialisiert. Diese gruppieren sich in einem "Active Space", in dem es genau einen Koordinator gibt. Dieser besitzt das Netzwerkschlüsselpaar, dessen öffentlicher Teil OOB an die übrigen Teilnehmer verteilt werden muss. Daraufhin ist eine vermittelte, verschlüsselte Kommunikation möglich.

Virtus ist eine Middleware, die in elektronischen Bezahlsystemen, im industriellen Kontext oder im eHealth-Sektor zum Einsatz kommen soll ([CBB+12]). Virtus basiert auf offenen und weit verbreiteten Technologien. So setzt es zur Kommunikation auf das Protokoll XMPP ([SA04]). Das hat zur Folge, dass die Kommunikation zwischen Teilnehmern zentralisiert stattfindet. Authentizität, Integrität und Vertraulichkeit werden durch TLS sichergestellt. Autorisierung wird in XMPP durch Zugriffslisten ("Access Control Lists", ACL) geregelt.

Neben den bisher genannten Arbeiten wurden weitere untersucht, bei denen Security nicht betrachtet wurde. Diese sollen an dieser Stelle lediglich genannt werden. Es handelt sich dabei um GREEN [SBC05], MobiPADS [CC03], MundoCore [AKM07], iCOCOA [BGI07], iLAND [vSS10], Cooltown [BK01, DGV03], PEIS [SB05, SBG+08, BGS+06, BRBS07], PACE [HIMB05], Uranus [Cor12], MUSDAC [CRI07], AURA [DOP11], SATware [HJJ+07], MidCASE [BJH+07] und ROOMWARE [TSP02].

6.1.5 Diskussion

Der Stand der Technik ist in den Tabellen 6.1 und 6.2 zusammengefasst. Das Sicherheitskonzept von DPWS selbst ist durch den Profilmechanismus sehr vielseitig. Die beschriebenen Profile bedienen sich jedoch für ressourcenschwache Systeme ungeeigneter Konzepte wie TLS und damit verbundenen X.509-Zertifikaten oder sind angreifbar. Während UPnP und DLNA keine Maßnahmen zur Wahrung der grundlegenden Schutzziele ergreifen, sind die Konzepte von Bluetooth, ZigBee und 6LoWPAN ihrer Natur nach für wenige schwache Teilnehmer innerhalb eines Subnetzes ausgelegt. Hierbei wird die Sicherheit zugunsten der Bedienbarkeit oft stark eingeschränkt, indem Bluetooth z.B. auf unveränderliche Pins zurückfällt oder netzwerkweite Schlüssel bei ZigBee inband und unverschlüsselt ausgetauscht werden, wodurch das gesamte Netzwerk als kompromittiert betrachtet werden muss. Schließlich sind die Sicherheitskonzepte untrennbar in die gesamten Protokollstapel verwoben, eine einfache Übertragung auf andere Technologien ist demnach nicht möglich.

Bei der Untersuchung verwandter Forschungsarbeiten in Abschnitt 6.1.4 stellte sich zunächst heraus, dass Sicherheit sehr häufig nicht oder unzureichend betrachtet wird. Von den insgesamt 22 betrachteten Technologien schlagen acht ein Sicherheitskonzept vor. Diese Tendenz wird auch deutlich, wenn man betrachtet, dass in einem Survey zu Kommunikationsmiddlewares von Saeed und Waheed ([SW10]) das Thema Kommunikationssicherheit allenfalls in einem Nebensatz erwähnt wird.

Die untersuchten Sicherheitskonzepte lassen sich in zwei Gruppen unterteilen. In der ersten Gruppe werden neue Sicherheitskonzepte vorgeschlagen und vollständig neue Lösungen entwickelt. Bei diesem Vorgehen entsteht das Problem, dass derartige Neuentwicklungen gründlich geprüft werden müssen. Dazu zählen Werkzeuge wie formale Analysen und Verifikation sowie Untersuchungen durch unabhängige Experten z.B. in Peer-Review-Prozessen. Das betrifft insbesondere den Neuentwurf auf kryptografischer und auf Protokollebene. Ein weiterer Nachteil liegt darin, dass die auf die jeweilige Technologie abgestimmte Sicherheitslösung selten Ansatzpunkte für interoperable Sicherheit zwischen verschiedenen Basistechnologien bietet.

Um die mit der Neuerfindung verbundenen zeitlichen und finanziellen Aufwände und Sicherheitsrisiken zu vermeiden, setzt die zweite Gruppe auf die

	UPnP & DLNA	Bluetooth	ZigBee	6LoWPAN	DPWS & Profile	MS HomeOS	Amigo	Hydra	SM4ALL	Plastic	PECES	Gaia	Mobile Gaia	Virtus
Dolev-Yao	□	☑	▣	▣	☑	□	▣	☑	☑	☑	☑	☑	☑	☑
Vermeidung von Single Point of Failures	□	☑	□	□	▣	□	□	▣	□	□	▣	□	□	▣
Kontinuierliches Deployment	□	☑	□	□	□	□	☑	□	☑	□	□	☑	☑	□
Externe Kommunikation	□	□	□	▣	☑	□	▣	☑	☑	☑	▣	□	□	☑
Flexible Autorisierung	□	□	□	□	□	▣	□	☑	▣	□	□	□	□	□
Administrative Gruppen	□	□	□	□	☑	□	▣	▣	▣	□	□	□	□	□
Sicherheitsstufen	□	☑	□	▣	☑	□	▣	☑	☑	▣	□	□	□	□
Skalierbar	□	□	□	▣	☑	□	▣	▣	☑	□	□	□	□	□
Benutzbarkeit	□	□	□	□	□	□	□	□	□	□	□	□	□	□
Unterstützung Datenhaltung	□	□	□	□	□	□	▣	☑	☑	☑	□	□	□	□

Tabelle 6.1: Vergleich der untersuchten Sicherheitslösungen und Zielstellung - Teil 1; ☑betrachtet — ▣z.T. betrachtet / möglich — □nicht betrachtet

Wiederverwendung existierender Mechanismen und Technologien wie TLS, X.509-Zertifikaten, nutzerzentrische Authentifizierung durch Nutzername und Passwort, der gesamten Web-Service-Security-Spezifikationssuite oder einzelnen Teilen, Kerberos oder RADIUS. Diese Mechanismen zählen zu Teilen weit verbreiteter Sicherheitsinfrastrukturen, die als modulare Bausteine in neuen Systemen genutzt werden können. Sie sind jedoch nicht für verteilte eingebettete Systeme entworfen worden, sondern zur Absicherung einzelner Verbindungen (TLS), in denen der Nutzer im Vordergrund steht (Nutzername und Passwort) bzw. für verteilte Systeme aus Desktop PCs und Web Servern (X.509-Zertifikat, PKI & WS Security). Für verteilte eingebettete Systeme existieren jedoch keine derart verbreiteten und bewährten Sicherheitsinfrastrukturen, die die Eigenarten einer stark inhomogenen Ressourcenverteilung in einem Netzwerk berücksichtigen.

Eine weitere Auffälligkeit bei der Untersuchung der Middlewares liegt im erkennbaren Übergang der Kommunikationsparadigmen. Während um die

	UPnP & DLNA	Bluetooth	ZigBee	6LoWPAN	DPWS & Profile	MS HomeOS	Amigo	Hydra	SM4ALL	Plastic	PECES	Gaia	Mobile Gaia	Virtus
Leistungsheterogenität	☐	☐	☑	☐	☐	☑	☐	☐	☐	☐	☐	☐	☑	☑
Entlastung Kleinstgeräte	☐	☐	☑	☐	☐	☑	☐	☐	☐	☐	☐	☐	☑	☑
Heterogenität Bedienelemente	☐	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Geräte im Fokus statt Nutzer	☐	☑	☑	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Schutz bei Verlust einzelner	☐	☐	☐	☐	☐	☐	☑	☑	☑	☐	☐	☐	☑	☐
Koordiniertes Abmelden	☐	☐	☐	☐	☐	☐	☑	☑	☑	☐	☐	☐	☑	☐
Wartung durch Experten und Nutzer	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Kleine Änderungen durch Nutzer	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Delegation von Rechten	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐
Weit verbreitete Technologie	☐	☑	☐	☑	☑	☐	☑	☑	☑	☑	☑	☐	☐	☐
Sichere Protokollinteroperabilität	☐	☐	☐	☐	☐	☑	☐	☐	☐	☐	☐	☐	☐	☐
Sichere Herstellerinteroperabilität	☐	☑	☐	☑	☐	☑	☑	☑	☑	☑	☐	☐	☐	☐
Interoperable Ende-zu-Ende Sicherheit	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐	☐

Tabelle 6.2: Vergleich der untersuchten Sicherheitslösungen und Zielstellung - Teil 2; ☑betrachtet — ☒z.T. betrachtet / möglich — ☐nicht betrachtet

Jahrtausendwende veröffentlichte Technologien sehr häufig auf objektorientierter Kommunikation wie CORBA basieren, stehen wenige Jahre später SOA als Paradigma und Web Services als konkrete Technologie im Vordergrund. Auch Petrie und Bussler konstatieren in [PB03], dass die Industrie die Verwendung von SOAP und WSDL – den grundlegenden Web Service Technologien – sehr begrüßt, wodurch sich eine hohe Akzeptanz und

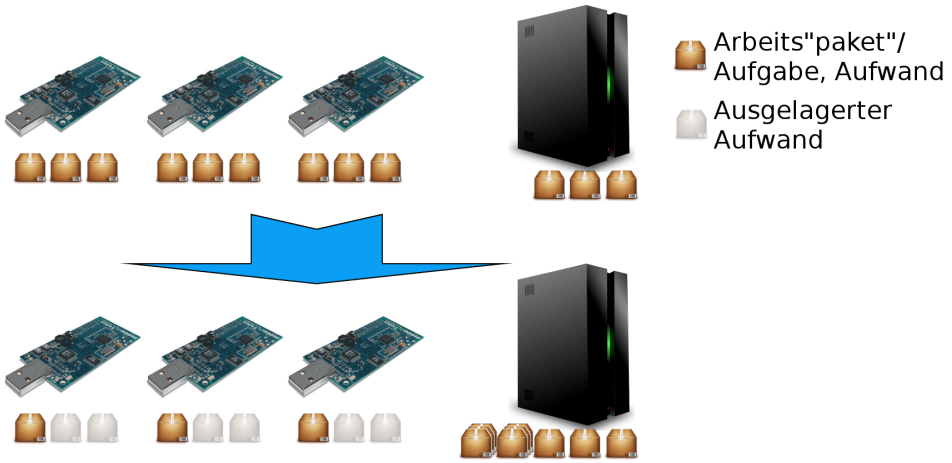
vorhandenes Know-How ableiten lassen. Beachtet man diesen Trend und berücksichtigt dabei die vorherrschende Intention, bestehende Sicherheitskonzepte wiederzuverwenden, so erklärt das die zahlreichen Empfehlungen, die Konzepte der Web Service Security Spezifikationssuite zu nutzen (z.B. [MLH+08, HLP+09, ERA09, BDCM+09, CRI10]).

Die Web Service Security Spezifikationssuite stellt eine umfassende Sicherheitsinfrastruktur für verteilte Systeme dar, die im geschäftlichen Umfeld weit verbreitet ist. Hinzu kommt, dass die Spezifikationen vollständig frei verfügbar sind. Die Zielplattformen für diese Spezifikationssuite sind jedoch Web Server und Desktop PCs, also Umgebungen, die grundsätzlich „ausreichend“ Ressourcen zur Verfügung haben. Ebenso wie die Basistechnologie Web Services mit der Schaffung von DPWS nun für den Einsatz auf eingebetteten Systemen zur Verfügung steht, werden in diesem Kapitel die Konzepte der Web Service Security Spezifikationssuite für den Einsatz auf eingebetteten Systemen umgesetzt. In Anlehnung an DPWS trägt die hier vorgestellte Lösung den Titel “Devices Profile for Web Services Security“ oder kurz DPWSec. Das Ziel von DPWSec ist es, eine Grundlage für eine interoperable, wiederverwendbare Sicherheitsinfrastruktur für verteilte eingebettete Systeme zu bieten, so wie WS Security, Kerberos und PKI Infrastrukturen für verteilte Desktop- und Serversysteme darstellen.

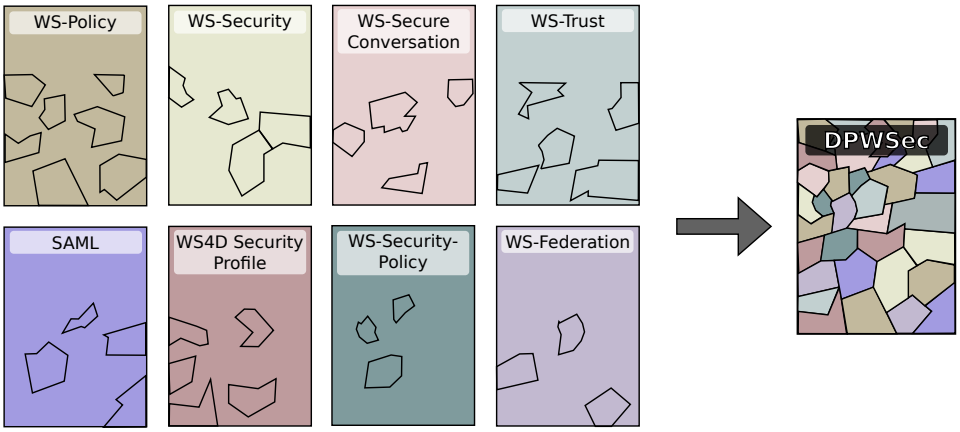
6.2 Methodik

Im folgenden Abschnitt wird die Methodik beschrieben, die in dieser Arbeit entwickelt wurde, um DPWSec aus der Gesamtheit der Web Service Security Spezifikationen herauszuarbeiten. Die Methodik besteht zunächst aus zwei übergeordneten Zielen, die schematisch in Abbildung 6.1 dargestellt sind.

Zunächst sollen im Rahmen der Möglichkeiten der Spezifikationen Wege gefunden werden, wie sicherheitsbezogene Aufwände von ressourcenschwachen auf stärkere Teilnehmer verlagert werden können (vgl. Abb. 6.1a). Es wird dabei vorausgesetzt, dass in jedem Netzwerk, das den Szenarien aus Kapitel 5.3 entspricht, „starke“ Geräte existieren, die über eine permanente Stromversorgung und „hinreichende“ Ressourcen hinsichtlich Speicher und Rechenkraft verfügen. Dabei muss jedoch sichergestellt werden, dass keine neuralgischen



(a) Auslagern von Aufwänden

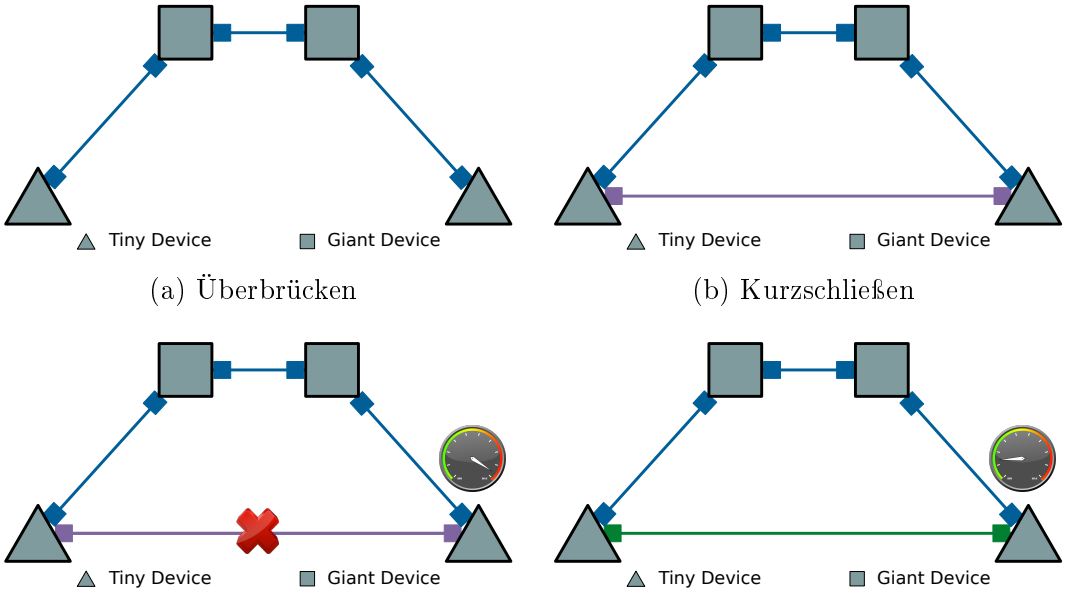


(b) Eliminieren von Spezifikationsteilen

Abbildung 6.1: Schema der entwickelten Methodik

Punkte – sogenannte Single Points of Failure – eingeführt werden. Der Ausfall eines einzelnen Geräts darf also nicht zum Ausfall der gesamten Kommunikation führen.

Dabei hat sich herausgestellt, dass sich insbesondere Aufgaben, die dem Aufbau von Vertrauensbeziehungen dienen, zur Auslagerung an Dritte eignen. Zu diesen Aufgaben zählen das Herunterladen und Parsen von Policies von Teilnehmern und anschließende Entscheidungsfindungen zur Nutzung von Authentifizierungsmechanismen. Hierzu zählt auch das Aushandeln von Verbindungsparametern wie die eingesetzten kryptografischen Algorithmen und Schlüssellängen. Außerdem können starke Teilnehmer bestehende Lücken in verketteten Vertrauensbeziehungen zunächst überbrücken (Abb. 6.2a) und an-



(a) Überbrücken (b) Kurzschließen

(c) Aufgeben (d) Automatisches Erneuern

Abbildung 6.2: Überbrücken, Kurzschließen, Aufgeben und automatisches Wiederherstellen von Vertrauensbeziehungen

schließend kurzschließen (Abb. 6.2b). Ein weiterer Punkt ist das Zwischenspeichern von Vertrauensbeziehungen. So können schwache Teilnehmer Daten zu einzelnen Vertrauensbeziehungen löschen, um ihre Speicherauslastung zu optimieren (Abb. 6.2c) und bei Bedarf über starke Teilnehmer für den Nutzer unbemerkt und transparent wiederherstellen (Abb. 6.2d). Neben Aufgaben der Authentifizierung können starke Teilnehmer zudem Autorisierungsaufgaben übernehmen, sodass schwache Teilnehmer keine aufwändigen Entscheidungsfindungsprozesse bewältigen müssen.

Neben dem Auslagern von Aufwänden soll der Umfang der Spezifikationen eingeschränkt werden, um die Spezifikationen selbst und somit ihre Implementierung zu vereinfachen (vgl. Abb. 6.1b). Dieses Einschränken von Funktionsumfängen, also das Bilden von Untermengen, ist der Vorgang der im Umfeld der Web Services als Bildung eines Profils verstanden wird. Ziel der Einschränkung sind Spezifikationsteile, die für eingebettete Systeme ungeeignet sind oder aufgrund der Anforderungsanalyse keine Anwendung finden. Während einer umfassenden Analyse der Spezifikationen haben sich Muster herauskristallisiert, die zu einer Richtlinie führten, welche Spezifikationsteile eliminiert werden können.

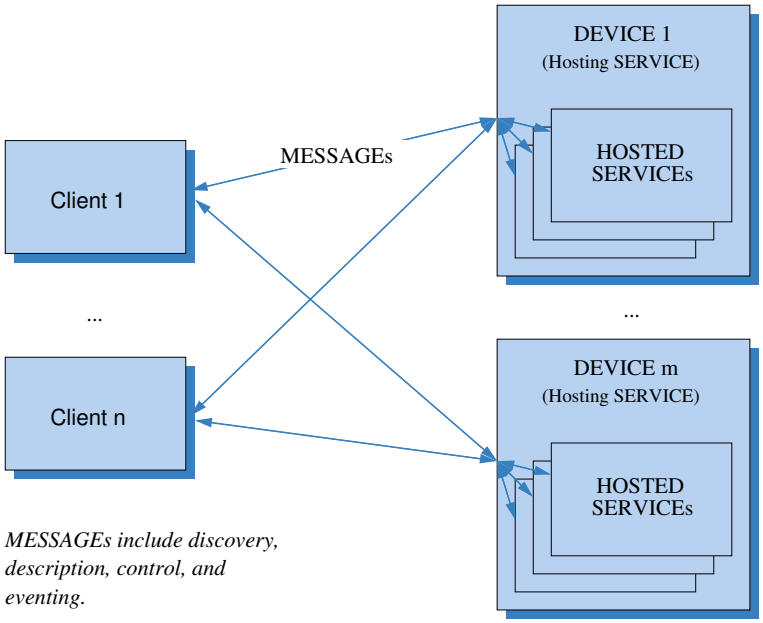
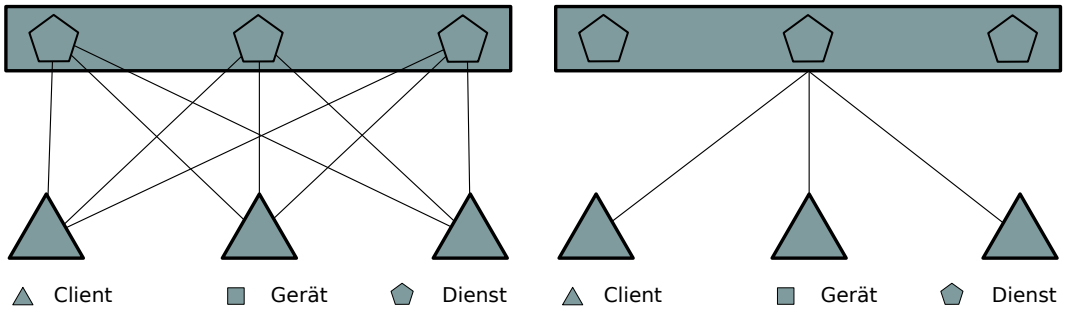


Abbildung 6.3: Kommunikationsmodell von DPWS nach [OAS09a]

Zunächst gilt es, das *Kommunikationsmodell von DPWS* zu betrachten, wie es in [OAS09a] definiert wird (vgl. Abb 6.3). Dieses Modell führt eine strikte Trennung ein zwischen Devices, die Dienste anbieten und Clients, die die Operationen dieser Dienste aufrufen. Das bedeutet, dass ein Device nicht notwendigerweise Client-Funktionalität zur Verfügung stellt und umgekehrt. Die Web Service Security Spezifikationssuite nimmt diese Trennung nicht vor. Sie setzt voraus, dass ein Gerät, dessen Dienste angesprochen wurden, anschließend als Client fungieren kann, um seinerseits Dienste anzusprechen, beispielsweise um Sicherheitstokens zu verifizieren. DPWSec hingegen behält die von DPWS eingeführte strikte Trennung bei. Sie ermöglicht es z.B. einem Device ohne Client-Funktionalität auszukommen, was die benötigten Bibliotheksteile reduziert und damit die Codegrößen möglichst gering hält.

Ein weiteres Muster lässt sich auf die Zusammenhänge aus Abbildung 6.3 zurückführen. In der Welt der Web Services ist ein Dienst eine isolierte, autonome Einheit und repräsentiert die höchste Stufe der strukturellen Hierarchie. In DPWS hingegen ist den Diensten eines Geräts das Gerät selbst übergeordnet, repräsentiert durch den *Hosting Service*. Während einzelne Dienste eines Geräts durchaus eigene Sicherheitsanforderungen stellen können, hängt die Unterstützung kryptografischer Algorithmen und Mechanismen von den Fähigkeiten



(a) Unabhängige Dienste (b) Verwaltung durch Gerät

Abbildung 6.4: Zu verwaltende Vertrauensbeziehungen

des Geräts ab. Des weiteren sollen alle Dienste eines Geräts eine gemeinsame Verwaltung der Vertrauensbeziehungen unterhalten (vgl. Abb. 6.4b). Würde jeder Dienst seine Vertrauensbeziehungen selbst verwalten, wäre der Mehraufwand beträchtlich, denn es müssten Authentifizierungsvorgänge für jede Client-Service-Kombination durchgeführt und die entstehenden Credentials gespeichert werden (vgl. Abb. 6.4a).

Weitere Vereinfachungen betreffen die Eigenschaft der *Zustandslosigkeit* der Hosted Services. Unter diesem Begriff wird die Tatsache verstanden, dass die Antwort eines Dienstes (bzw. einer seiner Operationen) in keiner Weise von vorangegangenen Anfragen an diesen Dienst abhängen darf. Masak bemerkt in [Mas09], dass diese Eigenschaft häufig nur schwierig vollständig umsetzbar ist. Ein guter Indikator ist demnach, wenn ein Dienst alle zur Bearbeitung der Anfrage notwendigen Daten als Parameter übergeben bekommt und nicht auf persistente Zusatzinformationen zurückgreifen muss. Einzige Ausnahme bildet hier der Zugriff auf einen weiteren Dienst zum Erlangen weiterer Daten. Die Zustandslosigkeit von Diensten gewährleistet die erwünschte lose Kopplung und damit die einfache Austauschbarkeit einzelner Dienste. Eine vollständig zustandslose Authentifizierung von Dienstaufrufen bedeutet jedoch, dass jeder Dienstaufruf eine vollständige Authentifizierung beinhaltet, inklusive dem Austausch aller Credentials (vgl. Abschnitt 6.3.5). Dies bedeutet jedoch einen enormen Mehraufwand in der Kommunikation. Schließlich würde die zuvor beschriebene Beziehung zwischen Diensten und Geräten unberücksichtigt bleiben. Auf Teile der Zustandslosigkeit zu verzichten bedeutet, Geräte weniger

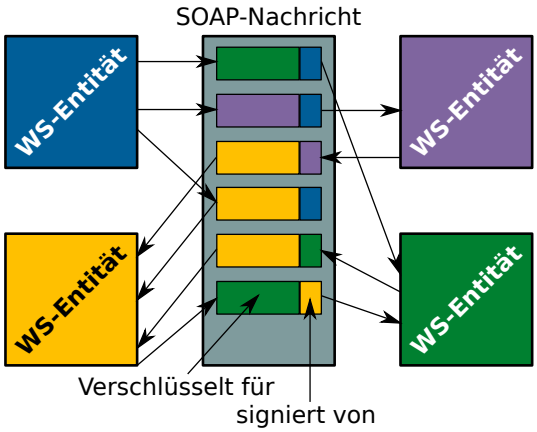


Abbildung 6.5: Schematische Darstellung von Multihop-Sicherheit

leicht austauschbar zu machen, vereinfacht jedoch das Kommunikationsmodell und verringert die Menge auszutauschender Daten.

Weiterhin lassen sich Spezifikationsteile eliminieren, die bestimmte Aspekte von Multihop-Sicherheit betreffen. Eine essentielle Anforderung besteht darin, dass Vermittler wie Router oder Gateways Nachrichten weiterleiten können müssen, ohne sie zu entschlüsseln. Die WS Security Spezifikationen bieten jedoch die Möglichkeit, diese Eigenschaft auch auf einzelne Teile einer Nachricht anzuwenden. Abbildung 6.5 zeigt dieses Vorgehen schematisch. So ist es einem Teilnehmer möglich, einer SOAP-Nachricht im Transit (verschlüsselte) Nachrichtenteile hinzuzufügen und separat zu signieren, ohne die Integrität der übrigen Nachrichtenteile zu beeinflussen. Um diese Funktionalität zu gewährleisten, müssen auf Senderseite alle verschlüsselten und signierten Nachrichtenteile einzeln referenziert werden. Auf Empfängerseite muss ein Parser einzelne Nachrichtenteile aus der SOAP-Nachricht isolieren und einzeln zu einem Digest zusammenfügen, um die Teilsignaturen bestätigen und die für ihn bestimmten Nachrichtenteile entschlüsseln zu können. Da in der Anforderungsanalyse keine Situation ausgemacht werden konnte, in der diese Funktionalität von Kleinstgeräten gefordert wird, wird im Profil DPWSec darauf verzichtet.

6.3 Das entstandene Profil und seine Funktionalität

Im vorangegangenen Abschnitt 6.2 wurde dargelegt, dass die Spezifikationen vereinfacht werden sollen, indem das Kommunikationsmodell von DPWS und die Gruppierung von Diensten auf einem Gerät durch den Hosting Service berücksichtigen werden. Weiterhin erfolgen Einschränkungen der Zustandslosigkeit der Dienste und der Multihop-Funktionalität. Ziel des Profils ist weiterhin, Aufwände von Authentifizierung und Autorisierung von schwachen auf starke Teilnehmer zu verteilen. Das entstandene Profil ist detailliert in Anhang B beschrieben. In diesem Abschnitt werden die Funktionalität und die entstehenden Abläufe dargelegt und werden mit den klassischen WS Security Spezifikationen verglichen.

6.3.1 Begrifflichkeiten

An dieser Stelle sollen Termini geklärt und zueinander in Bezug gebracht werden, die im restlichen Verlauf genutzt werden. Dies greift zum Teil einigen Konzepten vor, die im weiteren Verlauf erläutert werden, dient jedoch dem Zweck, die Lesbarkeit zu erhöhen.

Geräte ohne Ressourcenbeschränkung können vom Typ `Authenticator` und bzw. oder vom Typ `SynchronousAuthorizer` sein. In diesem Fall bieten sie einen speziellen Dienst an, den *Security Token Service (STS)*, der in WS Trust definiert wird. Ein Authenticator bietet die Funktionalität der delegierten Authentifizierung an. Ist ein Authenticator für ein Device oder einen Client zuständig, wird er als dessen *Broker* bezeichnet. SynchronousAuthorizer bieten in ihrem STS eine Operation zur Abfrage statisch konfigurierter Autorisierungen.

Multimedia-Geräte können vom Typ `UI-Authenticator` und bzw. oder vom Type `AsynchronousAuthorizer` sein. Auch sie beherbergen einen STS. Erstere dienen der Übertsetzung von OOB-Kanälen, wie in Kapitel 4 beschrieben. Letztere dienen der Abfrage von Nutzerentscheidungen hinsichtlich Autorisierungen.

Ein Client, der einen Authentifizierungs- oder einen Autorisierungsvorgang initiiert, wird als *Requestor* bezeichnet, das Ziel dieses Vorgangs als *Target*, bzw. im Kontext von Autorisierungen auch als *Subjekt*.

6.3.2 Allgemeine Erweiterungen

Zunächst werden allgemeine Erweiterungen und Änderungen besprochen, die nicht vordergründig die Sicherheit betreffen, jedoch für eine sichere Funktionsweise und für die Berücksichtigung von Kleinstgeräten von Belang sind.

Identifizierung

In der DPWS Spezifikation ([OAS09a]) ist geregelt, dass jedes DPWS-Device eine global eindeutige Identifikation besitzen muss, die persistent Reinitialisierungen des Geräts standhalten muss. Die Identifikation muss die Form eines Uniform Resource Name Universal Unique IDentifier (URN UUID) besitzen und somit das Format `urn:uuid:`*`[128-Bit-Adresse]`* aufweisen. Diese Identifikation wird insbesondere für die Adressierung des Devices genutzt. DPWSec verwendet die Identifikation weiterhin zur Zuweisung und Erkennung aufgebauter Vertrauensverhältnisse.

In DPWS ist es nicht notwendig, dass ein Client adressierbar ist und folgerichtig gibt es keinen Mechanismus für einen Client, eine URN UUID anzugeben. DPWSec verlangt hingegen eine Identifikation von Clients, um die Authentizität von Nachrichten überprüfen zu können. Daher wird festgelegt, dass Clients ebenfalls eine Identifikation besitzen, die den gleichen Ansprüchen genügen muss wie denen für Devices. Clients übermitteln ihre URN UUID während einer Authentifizierung. Das Ergebnis dieser Authentifizierung ist ein Security Context Token (SCT), das bei folgenden Dienstaufrufen als Tokenreferenz übermittelt wird, wodurch die Client-Identifizierung möglich wird. Die Verwendung des SCT und seine Entstehung während einer Authentifizierung werden in den folgenden Abschnitten 6.3.4 und 6.3.5 näher beleuchtet.

Ressourcensparendes Discovery

Das Auffinden von Geräten in einem Netzwerk – das sogenannte *Discovery* – funktioniert, indem der suchende Client eine Anfrage an die Multicast-Adresse des Netzwerks versendet. Alle auf diese Adresse lauschenden Geräte empfangen diese Nachricht. Gesucht werden können so alle anwesenden Geräte im Netzwerk oder alle Geräte einer bestimmten Typenkombination. Weitere Möglichkeiten zur Filterung der Ergebnisse sind nicht vorgesehen.

Häufig ist eine Antwort auf Suchanfragen nur von zuvor authentifizierten Geräten von Interesse. Um die Ressourcen der übrigen Geräte zu schonen und anzuzeigen, dass sie auf Grund eines fehlenden Vertrauensverhältnisses nicht auf die Suchanfrage antworten müssen, werden in dieser Arbeit die Begriffe des Low-Effort- und High-Effort-Discovery eingeführt. Bei ersterem sollen nur bereits authentifizierte Geräte antworten, bei letzterem alle.

Um dieses Verhalten zu gewährleisten, kann ein Client seine Suchnachricht mit einem speziellen Header-Feld versehen, um anzuzeigen, dass es nur authentifizierte Geräte sucht. Darüber hinaus muss ein Client seine Identifikation in das Header-Feld schreiben, das die Antwortadresse enthält. Das verletzt im geringen Maße die Spezifikationen von DPWS und WS Discovery. Eine ausführliche Diskussion dessen findet sich in Anhang B.6 und zeigt, dass dies nur Geräte betrifft, die DPWSec unterstützen, sodass eine Abwärtskompatibilität gegeben ist.

Zweistufiges Discovery & Description

Nachdem ein Device durch den Discovery-Prozess aufgefunden wurde, muss seine Beschreibung heruntergeladen werden, die Angaben zum Gerät selbst sowie Arten, Namen und Adressen der gehosteten Dienste enthält. Weitere Angaben zu den Diensten, wie z.B. die angebotenen Operationen, halten die Beschreibungen der jeweiligen Dienste bereit. Die eigentliche Authentifizierung des Geräts kann erst nach dem Description-Vorgang stattfinden. Somit sind Discovery- und Description-Vorgänge als unsicher zu betrachten, da grundsätzlich alle Angaben ungeschützt waren.

Aus diesem Grund müssen die beiden Phasen nach erfolgreicher Authentifizierung wiederholt werden. Dabei müssen Devices alle Antworten signieren, damit der anfragende Client die Angaben verifizieren kann. Damit das Device erkennt, dass es und für welchen Client es seine Antworten signieren muss, geben Clients ihre Identifikation bei der Rücksendeadresse an und signieren ebenfalls ihre Anfragen. Außerdem geben Clients die Identifikation des Devices im Empfängerfeld an, damit ausschließlich das Gerät antwortet, mit dem die abgesicherten Phasen durchgeführt werden sollen. Besteht eine Diskrepanz zwischen den Angaben von unsicherem und sicherem Discovery oder Descrip-

tion, müssen alle Angaben verworfen und die Vertrauensbeziehung umgehend aufgelöst werden.

Geräteklassen und -typen

Um den heterogenen Fähigkeiten der teilnehmenden Geräte in einem Netzwerk Rechnung zu tragen, werden in dieser Arbeit verschiedene Geräteklassen eingeführt. Zunächst erfolgt die Unterscheidung zwischen „großen“ (*Giant*) und „kleinen“ (*Tiny*) Geräten. Sie stellen die Klassen für Geräte ohne beziehungsweise mit starken Ressourcenbeschränkungen dar. Im Kontext von DPWSec bedeutet es, dass Geräte der Giant-Klasse Aufwände von Geräten der Tiny-Klasse übernehmen. In DPWSec wird bewusst keine konkrete Grenze zwischen Tiny und Giant quantifiziert, z.B. auf Basis der Taktfrequenz, denn diese kann den Gegebenheiten des Szenarios angepasst werden. Indikatoren sind neben Taktfrequenz auch zur Verfügung stehender Speicherplatz oder Energieversorgung. So ließen sich mobile Geräte wie drahtlose Sensoren, deren Batterien nicht hochfrequent getauscht werden sollen, der Tiny-Klasse zuordnen, um die Batterien zu schonen. Alle Geräte mit (quasi)permanenter Stromversorgung, wie Steuerungsserver, PCs oder Router übernehmen dann die Aufgaben der Giant-Klasse.

Zusätzlich zu den „kleinen“ und „großen“ Geräten wird eine dritte Klasse eingeführt, die UI-Geräte. UI steht für User Interface und diese Klasse repräsentiert vor allem Multimediageräte, die direkt vom Nutzer bedient werden, teilweise sogar vom Nutzer mitgeführt werden. Dadurch besitzen sie häufig eine hohe Anzahl an Ein- und Ausgabemöglichkeiten. Ihre Stromversorgung ist nicht notwendigerweise permanent, jedoch spielt die Schonung ihrer Ressourcen eine untergeordnete Rolle. Beispiele für diese Geräteklasse sind Smartphones, Smart-TVs, digitale Bilderrahmen und Wearables wie Smartwatches oder Aktivitätstracker.

6.3.3 Sicherheit auf Nachrichtenebene

In der Web Service Security Spezifikationssuite ist die Basisspezifikation WS Security verantwortlich für die Gewährleistung der Sicherheit auf Nachrichtenebene. Das beinhaltet Vertraulichkeit, Integrität und Authentizität einer Nach-

richt bei bekanntem Schlüsselmaterial. Diese werden bei DPWSec durch das WS Compact Security Schema bereitgestellt, das ausführlich in Kapitel 3.2 beschrieben wurde.

Darüber hinaus werden aus der WS Security Spezifikation die Abschnitte zu ID-Referenzen (Kapitel 4) und dem Security Header (Kapitel 5) übernommen. ID-Referenzen stellen einen Mechanismus dar, um einzelne Elemente innerhalb einer SOAP-Nachricht durch ein Attribut mit dem Namen `id` zu referenzieren. Die Definition des Security-Headers beinhaltet lediglich ein SOAP-Header-Element mit dem Namen `<Security>`. Außerdem wird in Kapitel 10 ein Header-Feld für Timestamps definiert, in dem Erstellungs- und Ablaufdatum und -zeit eines Vertrauensverhältnisses angegeben werden können. Auch dieser Abschnitt wird unverändert in DPWSec übernommen. Um die eigentliche Message Freshness zu gewährleisten, und somit Replay-Angriffen zu begegnen, wird der in DPWS bzw. in WS Addressing ohnehin vorgesehene Mechanismus der MessageIDs verwendet.

6.3.4 Sicherheit auf Verbindungsebene

Um ein aufgebautes Vertrauensverhältnis zwischen zwei Parteien zu kommunizieren, z.B. am Ende einer direkten Authentifizierung oder bei der Vermittlung von Vertrauensverhältnissen, bedarf es eines speziellen Mechanismus. Dieser wird in WS Secure Conversation durch das *Security Context Token* (SCT) bereitgestellt. Neben dem Format für SCTs werden aus WS Secure Conversation die Möglichkeiten übernommen, einen bestehenden Sicherheitskontext zu erneuern oder für ungültig zu erklären.

Die Spezifikationen WS Security und WS Secure Conversation sehen vor, dass in jede verschickte SOAP-Nachricht der genutzte Sicherheitskontext in Form eines SCT eingebettet wird. Da jedoch zwischen zwei Geräten immer genau ein Kontext aufgebaut wird, genügt es, Sender- und Empfänger IDs anzugeben. Auf diese Weise kann – unter Verlust eines gewissen Grades an Zustandslosigkeit – implizit der korrekte gespeicherte Kontext durch die Geräte ausgewählt werden. Die Angabe des Kontextes erfolgt durch die `KeyId`-Felder, die WS Compact Security (vgl. Kapitel 3.2, bzw. Listing 3.2) bereitstellt. Neben der Kontextreferenz können diese beiden Felder Angaben zu eventuell abgeleiteten Schlüsseln enthalten, die zu verwendende Key Deriviation Func-

tion ist dabei während des Verbindungsaufbaus ausgehandelt worden. Somit ist die in Abschnitt 7 von WS Secure Conversation definierte Weise zur Angabe abgeleiteter Schlüssel in nicht DPWSec notwendig.

6.3.5 Authentifizierung

Der Entwurfsmethodik aus Abschnitt 6.2 folgend, wurde in dieser Arbeit eine Authentifizierungsinfrastruktur entwickelt, die auf WS Trust basiert, dabei jedoch die Besonderheiten verteilter eingebetteter Systeme berücksichtigt, um so den Anforderungen aus Kapitel 5.4 gerecht zu werden. Im folgenden Abschnitt wird kurz zusammengefasst, wie das Trust-Brokering – also die nahtlose und transparente Vermittlung von Vertrauensverhältnissen – durch WS Trust vorgegeben wird. Anschließend werden Nachteile dieses Vorgehens aufgezeigt und es wird eine Alternative hergeleitet, die die Möglichkeiten von WS Trust nutzt und gleichzeitig den Anforderungen aus den Abschnitten 6.2 und 5.4 genügt. Da die reine Vermittlung von Vertrauensverhältnissen keiner umfassenden Authentifizierungsinfrastruktur genügt, wird zudem erläutert, wie sich eine direkte Authentifizierung in das vorgeschlagene Konzept einfügt. Die vollständige Lösung wird dann in Form von Fluss- und Kommunikationsdiagrammen beschrieben.

Die Delegation von Vertrauensverhältnissen ohne Public-Key-Infrastrukturen wird durch WS Trust folgendermaßen bewerkstelligt (vgl. Abb. 6.6). Vorausgesetzt wird, dass der Client aufgrund vorhergehender Discovery- und Description-Vorgänge eine Anfrage an den Service richten möchte und dass ihm bekannt ist, dass er dazu ein Token des STS benötigt. Außerdem besteht eine Vertrauensbeziehung zwischen Service und STS und die Credentials des Clients (z.B. eine Nutzername-Passwort-Kombination) sind in der Datenbank eingetragen. Der Client richtet unter Angabe seiner Credentials eine Anfrage an den STS, in der er eine Authentifizierung gegenüber dem Service beantragt. Der STS überprüft durch eine Datenbank die angegebenen Credentials und erstellt im Erfolgsfall ein Token, das es dem Client ermöglicht, mit dem Service zu kommunizieren. Dieses Token enthält die Identifikation des Clients, des Services sowie Verbindungsparameter wie zu verwendende Verschlüsselungsalgorithmen und Schlüssellängen. Das zugehörige *Proof-of-Possession*-Token (PoP-Token) enthält das Schlüsselmaterial für die Verbindung zwischen Client

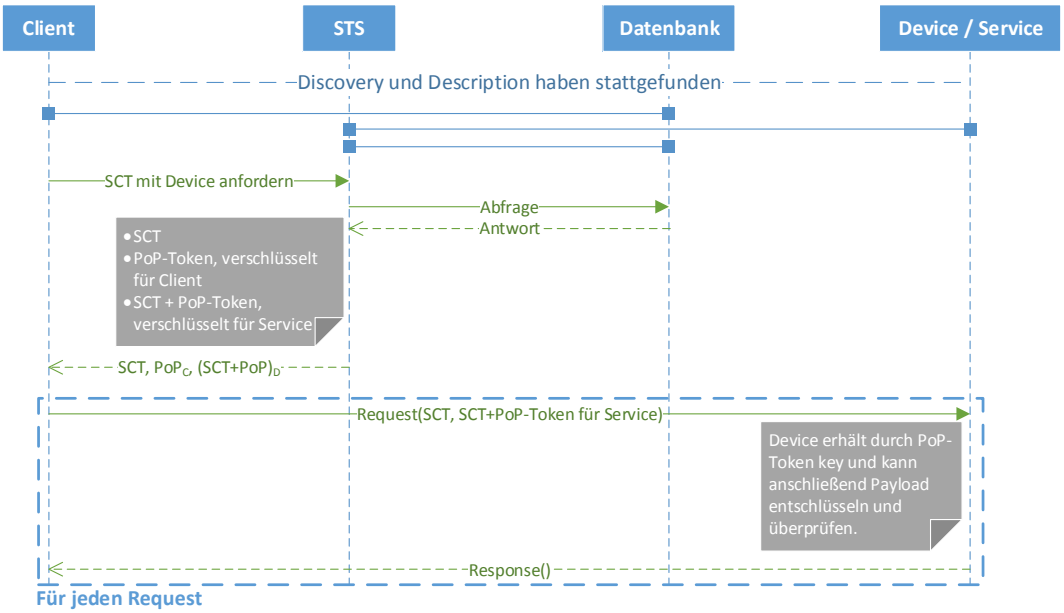


Abbildung 6.6: Trust-Delegation in WS Trust

und Service. Die Antwort beinhaltet demnach sowohl das SCT als auch das PoP-Token, jeweils verschlüsselt für den Service und für den Client.

Die nun folgenden Requests an den Dienst enthalten das SCT, das den Verbindungskontext mit seinen Parametern enthält. Damit der Dienst dessen Verlässlichkeit überprüfen kann, müssen die Requests außerdem die vom STS für den Dienst verschlüsselten und signierten Tokens enthalten. Der Dienst kann dessen Echtheit durch die Überprüfung der Signatur des STS überprüfen und den darin eingebetteten verschlüsselten Schlüssel für die gesicherte Kommunikation zwischen Client und Service nutzen. Um zu gewährleisten, dass die Zustandslosigkeit erhalten bleibt, muss das Token in jeden Request eingebettet werden, was zu großen Nachrichten und hohem Verkehrsaufkommen führt. Zudem muss der Service die empfangenen Nachrichten aufwändig parsen, um die verschiedenen Hashwerte, die zur Überprüfung der verschiedenen Signaturen in einer Nachricht notwendig sind, bilden zu können.

In Abbildung 6.7 ist zu erkennen, dass die Vermittlung von Vertrauensverhältnissen mit WS Trust gut skaliert, denn eine Vertrauensbeziehung kann prinzipiell über beliebig viele Security Token Services hinweg delegiert werden. Auf diese Weise lassen sich Vertrauenshierarchien aufbauen (vgl. Abb.2.2, S. 19) oder Trust Domains miteinander verknüpfen (vgl. Abb.2.6 c), S. 35).

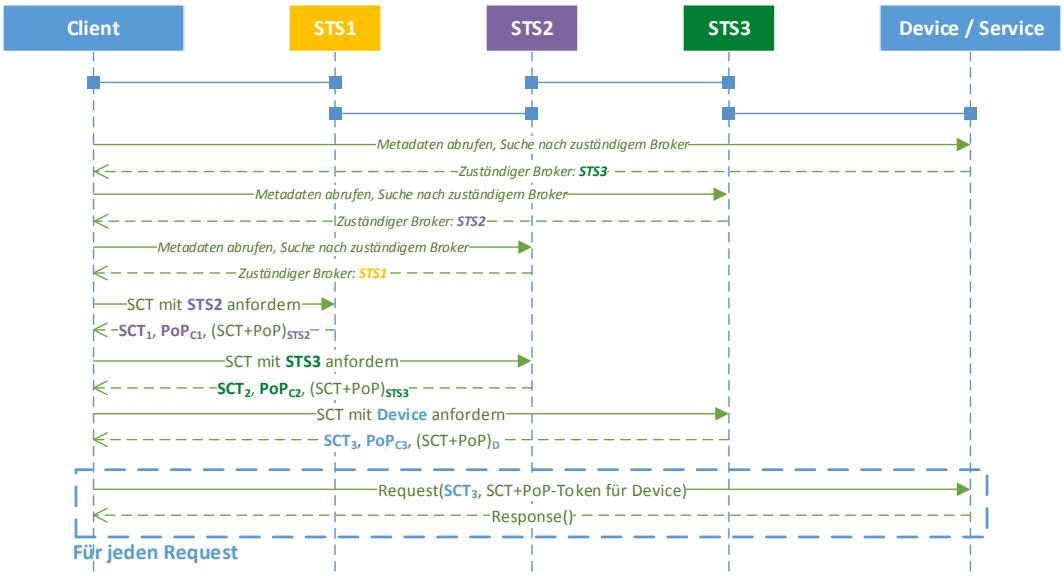


Abbildung 6.7: Trustdelegation in WS Trust über mehrere Security Token Services

Es ist jedoch erkennbar, dass der Aufwand für den Client dabei sehr hoch ist. Er muss eigenständig den Pfad über verschiedene Security Token Services auflösen und mehrere Authentifizierungen vornehmen, um ein Token zur Kommunikation mit seinem eigentlichen Ziel zu erhalten. Um einen potentiell ressourcenschwachen Client zu entlasten, wäre es demnach wünschenswert, wenn dieser sich an STS1 wenden könnte, mit der Aufforderung, ihm ein Token für den eigentlichen Service auszustellen. Anschließend übernimmt STS1 die Auflösung des Trust-Pfades und stellt dem Client ein SCT und ein PoP-Token aus. Ein mögliches Verfahren ist in Abbildung 6.8 dargestellt. Auf diese Weise übernehmen die stärkeren Teilnehmer, die die Security Token Services hosten, einen erheblichen Teil des Aufwandes während der Authentifizierung.

In den nun möglichen Requests ist es die Verantwortung des Clients, das Security Context Token sowie die verschlüsselten Teile für den Service anzuhängen. Um das so entstehende Datenvolumen zu reduzieren, wird die Anforderung an die Eigenschaft der Zustandslosigkeit eingeschränkt. Um das SCT und die verschlüsselten Teile für den Empfänger nicht an jeden Request anhängen zu müssen, dürfen sie nach dem ersten Austausch zwischengespeichert werden. Das eigentliche SCT kann durch eine kompakte Token-Referenz aus-

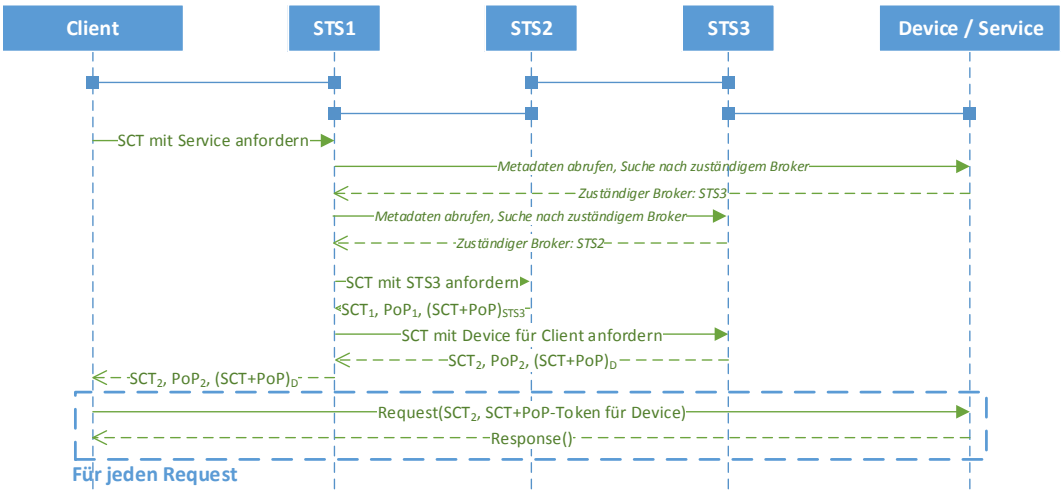


Abbildung 6.8: Alternative Trust-Delegation mit Verlagerung des Aufwandes zur Auflösung des Trust-Pfades

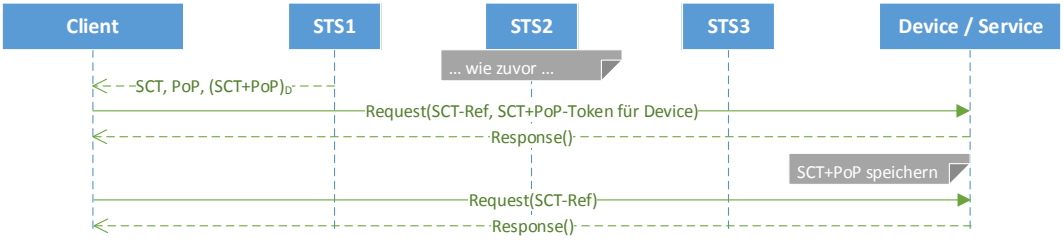


Abbildung 6.9: Nutzung von Token-Referenzen zur Reduzierung des Datenverkehrs

getauscht werden und sowohl Client als auch Service sind dafür zuständig, die eigentlichen Kontext-Token bei Eintreffen einer Token-Referenz aufzulösen (vgl. Abb. 6.9).

Dennoch besteht bei diesem Vorgehen ein Ungleichgewicht hinsichtlich der Aufwände bei Client und Service. Der Client ist noch immer dafür zuständig, die für den Service verschlüsselten Teile entgegenzunehmen und ihn weiterzuleiten, was zusätzlichen Verwaltungs- und Kommunikationsaufwand bedeutet. Um diesen Mehraufwand für den Client zu vermeiden und damit auch den Gesamtaufwand zu minimieren, könnte der Service die Referenz selbstständig auflösen, wie es in Abbildung 6.10 dargestellt ist. Das verletzt jedoch das Kommunikationsmodell von DPWS, da dadurch Client-Funktionalität von einem Dienst verlangt wird. Aus diesem Grund sieht DPWSec eine Service-

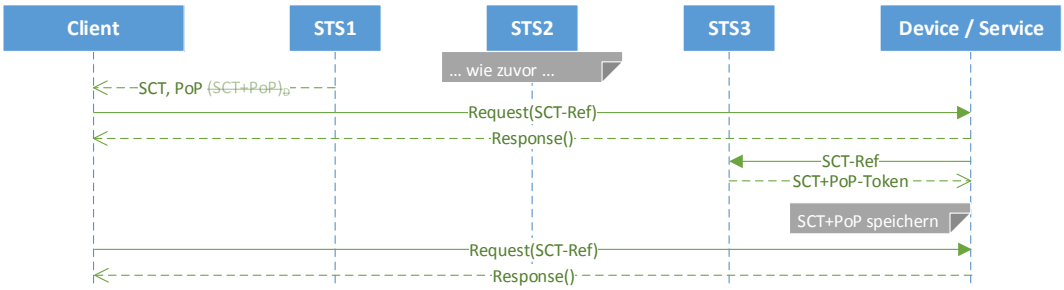


Abbildung 6.10: Gleichverteilung des Aufwands durch dynamisches Auflösen von Token-Referenzen

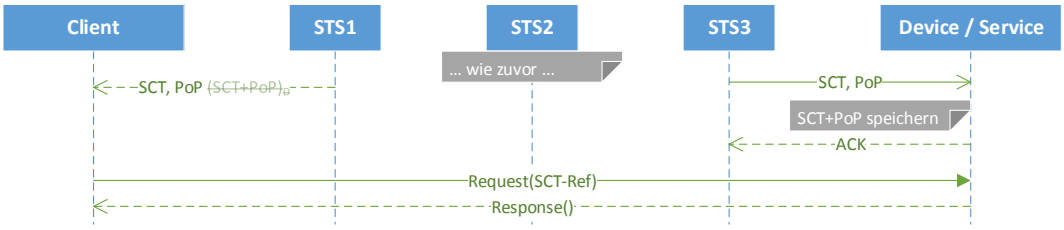


Abbildung 6.11: Asynchrone Auslieferung von Sicherheitskontexten

Schnittstelle für Devices vor, um asynchron Sicherheitskontexte von vertrauten Trust-Brokern entgegenzunehmen. Dieser Vorgang ist in Abbildung 6.11 dargestellt.

Durch das beschriebene Verfahren werden folgende Punkte erreicht. Zunächst ist der Aufwand für den Client beim Aufbau einer delegierten Vertrauensbeziehung minimiert. Er richtet lediglich eine Anfrage an einen Security Token Service, dem er vertraut, seinem sog. *Broker*. Diese Anfrage enthält das Authentifizierungsziel sowie unterstützte Verbindungsparameter. Der STS übernimmt die Auflösung des Trust-Pfades und bestimmt die Verbindungsparameter. Daraus erstellt er je ein Sicherheitskontext-Token für den Client und für das Device. Anschließend ist eine direkte Kommunikation für beide unter Angabe einer Token-Referenz möglich, ohne das SCT übertragen zu müssen, was die Nachrichtengrößen reduziert. Das beschriebene Verfahren trägt vor allem der Heterogenität der Geräte Rechnung, da potentiell ressourcenarme Teilnehmer ihre Aufwände auf die stärkeren verlagern können.

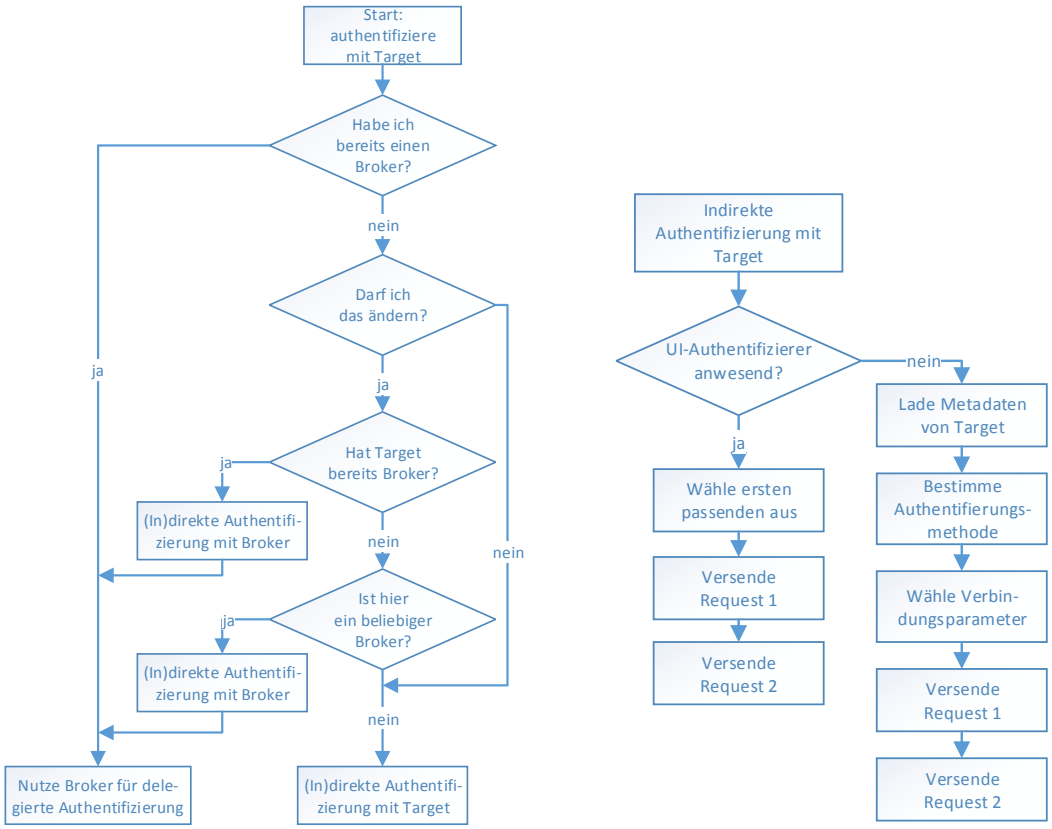
Wie eingangs bereits erwähnt, ist die delegierte Authentifizierung nur ein Aspekt einer vollständigen Authentifizierungsinfrastruktur. Vervollstän-

digt wird das Konzept durch die Möglichkeit, Vertrauensbeziehungen zwischen zwei Teilnehmern direkt aufzubauen. Nur so können sichere Verbindungen z.B. zwischen Client oder Device und einem STS aufgebaut werden. Auch Security Token Services benötigen eine Möglichkeit, dynamisch Vertrauensverhältnisse untereinander aufzubauen. In DPWSec wird hierzu der Gerätetyp des Authentifizierungsendpunkts eingeführt (vgl. Anhang B.3). Ein Gerät dieses Typs besitzt eine definierte Schnittstelle zum Aufbau einer Vertrauensbeziehung durch die Teilnahme an einem authentifizierten Elliptic-Curve-Diffie-Hellman-Protokolls, wie es bereits in Kapitel 4.2.2 beschrieben wurde. Dabei einigen sich beide Teilnehmer auf die notwendigen Verbindungsparameter wie zu verwendende Algorithmen und Schlüssellängen und leiten das gemeinsame Schlüsselmaterial ab. Authentifiziert wird dieser Verbindungsaufbau über eine Einweg-Pin, die Out-of-Band ausgetauscht wird. Auch das Konzept der indirekten Authentifizierung, dem sich Kapitel 4 dieser Arbeit widmet, geht vollständig in DPWSec ein. Das vereinfacht die sichere Benutzung und berücksichtigt die Heterogenität bzw. starke Einschränkung der Bedienelemente der verschiedenen Geräte.

Aufgaben der einzelnen Geräteklassen

Die beschriebenen Teillösungen lassen sich so miteinander kombinieren, dass eine robuste Authentifizierungsinfrastruktur für verteilte eingebettete Systeme entsteht. Ihre starken Teilnehmer entlasten dabei die schwächeren, wobei kein Single-Point-of-Failure entsteht, da grundsätzlich mehrere Security Token Services in einem Netzwerk parallel zueinander und miteinander agieren können. Um Kleinstgeräte zu entlasten, werden sowohl delegierte als auch (in)direkte Authentifizierung genutzt. Für den Fall, dass diese Funktionalitäten nicht zur Verfügung stehen, können die Kleinstgeräte mit entsprechendem Mehraufwand darauf verzichten und die Aufgaben vollständig dezentral selbst übernehmen. Im Folgenden werden die Aufgaben und internen Abläufe der verschiedenen Geräteklassen im Einzelnen beschrieben. Anschließend wird ihr Zusammenspiel in verschiedenen Authentifizierungsszenarien dargestellt.

Die Vorgänge in einem ressourcenbeschränkten Client sind in Abbildung 6.12 dargestellt. Wenn ein Client zum ersten Mal mit einem Device kommunizieren möchte, so ist er selbst noch keinem eigenen Broker zugeordnet (vgl.



(a) Entscheidung zwischen delegierter und (in)direkter Authentifizierung

(b) (In)direkte Authentifizierung

Abbildung 6.12: Vorgänge in einem ressourcenbeschränkten Client während einer Authentifizierung

Abb. 6.12a). Sofern es ihm nicht untersagt ist, sich einen Broker zu suchen, prüft er zunächst, ob das Target, mit dem er sich authentifizieren möchte, bereits einen Broker besitzt. In diesem Fall wählt er diesen als seinen eigenen Broker, alternativ wählt er einen beliebigen im Netzwerk. Mit dem gewählten Broker führt der Client nun eine (in)direkte Authentifizierung durch. Anschließend kann der Client den eben authentifizierten Broker für die delegierte Authentifizierung mit dem Target nutzen. Dies gilt auch für jeden folgenden Authentifizierungsvorgang, ohne, dass eine erneute (in)direkte Authentifizierung des Clients notwendig wird.

Der Mechanismus der indirekten Authentifizierung, bei der ein Multimediagerät die Übersetzung von OOB-Kanälen ineinander übernimmt, wurde in Kapitel 4 ausführlich beschrieben. Unter dem Begriff *(in)direkte Authentifizierung*

(vgl. Abb. 6.12b) werden nun die sich ergebenden Möglichkeiten zusammengefasst, für die Fälle, dass ein solches Multimediagerät zum Zeitpunkt der Authentifizierung vorhanden ist oder nicht. Im ersten Fall wendet sich der Client an den ersten UI-Authentifizierer, der einen seiner Mechanismen unterstützt und fordert eine Authentifizierung mit dem Target an. Der UI-Authentifizierer ermittelt anschließend die notwendigen Verbindungsparameter und wählt eine Authentifizierungsmethode mit dem Target aus. Für den Fall, dass der letzte Schritt nicht möglich ist, wurde in Kapitel 4.5.1 die Kaskadierung mehrerer UI-Authentifizierer beschrieben. Ist kein UI-Authentifizierer vorhanden oder im Falle des endgültigen Fehlschlagens einer indirekten Authentifizierung, nimmt der Client die Authentifizierung mit dem Target direkt vor. Das beinhaltet auch die Aushandlung aller Verbindungsparameter und den damit verbundenen Mehraufwand. Ist eine direkte Authentifizierung aufgrund inkompatibler OOB-Kanäle ebenfalls nicht möglich, so schlägt der Verbindungsaufbau fehl.

Die Abläufe innerhalb eines UI-Authenticators während der indirekten Authentifizierung sind in Abbildung 4.15c in Kapitel 4.6 beschrieben. Abbildung 6.13 zeigt hingegen die Vorgänge innerhalb eines Authenticators, wenn er eine Anfrage zur Durchführung einer delegierten Authentifizierung mit einem Target erhält. Zunächst wird überprüft, ob eine Vertrauensbeziehung mit dem Anfragesteller (dem sog. Requestor) besteht, da die Delegation von Trust-Verhältnissen nur für authentifizierte Teilnehmer möglich ist. Im einfachsten Fall ist der angesprochene Authenticator auch ein eingetragener Broker für das Target. Dann kann der Authenticator einen Sicherheitskontext generieren, der neben den Identifikationen der Teilnehmer die zu verwendenden Algorithmen, Schlüssellängen und das Schlüsselmaterial enthält und als SCT sowohl an den Requestor als auch an das Target ausliefern. Ist das Target dem Netzwerk gerade erst beigetreten, so ist ihm noch kein Broker zugewiesen. Der angesprochene Authenticator initiiert daher zunächst eine (in)direkte Authentifizierung mit dem Target und bietet sich dabei als Broker an. Anschließend fährt er mit der Generierung und Auslieferung eines SCT fort.

Im aufwändigsten Fall besitzt das Target bereits einen anderen Broker. Falls der angesprochene Authenticator diesem nicht vertraut, baut er zunächst eine Vertrauensbeziehung mit ihm auf. Dies kann delegiert oder (in)direkt geschehen und läuft ebenfalls so ab, wie hier beschrieben. Anschließend leitet er die

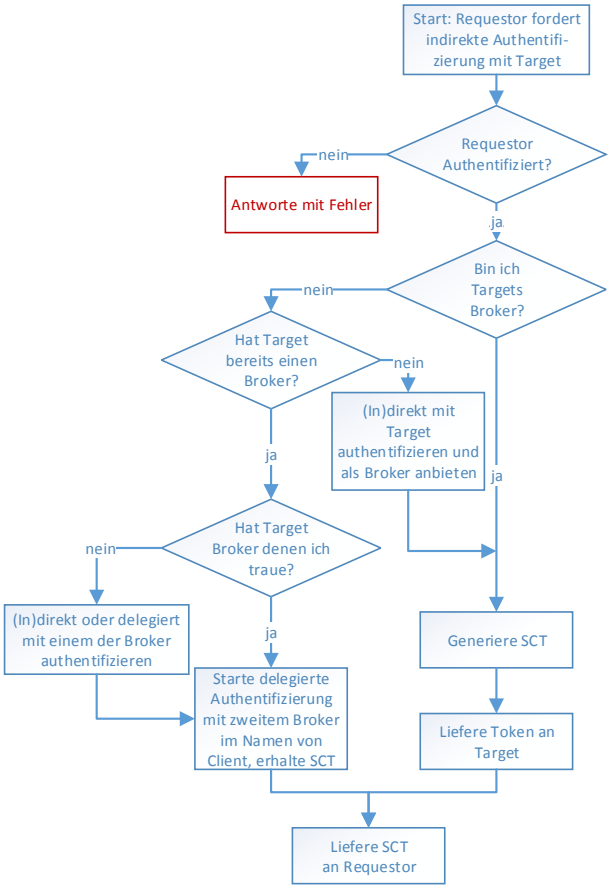


Abbildung 6.13: Aufgaben des Brokers während einer delegierten Authentifizierung

Anfrage zur delegierten Authentifizierung an den Broker des Targets weiter. Der Broker der Targets durchläuft dieselben Schritte und stellt dem angesprochenen Authenticator schließlich ein SCT aus, das er an den Requestor weiterleitet. Damit ist auch eine kaskadierte, delegierte Authentifizierung über mehrere Security Token Services hinweg möglich, ähnlich, wie es zu Beginn des Abschnitts in Abbildung 6.7 dargestellt ist.

Schließlich werden die Abläufe innerhalb eines Authentifizierungsendpunkts beschrieben. Diese Geräteklasse betrifft alle Devices und verlangt keine Client-Funktionalität. Aus Abbildung 6.14 gehen zwei Aspekte hervor. Zunächst ist wichtig, dass Geräte der Authenticator-Klasse während einer Authentifizierung angeben, dass sie sich als Broker anbieten. So lassen sich Authentifizierungsendpunkte einen oder mehrere Broker zuweisen und sie propagieren deren Adressen in ihren Metadaten, damit Clients diese Adressen herausfin-

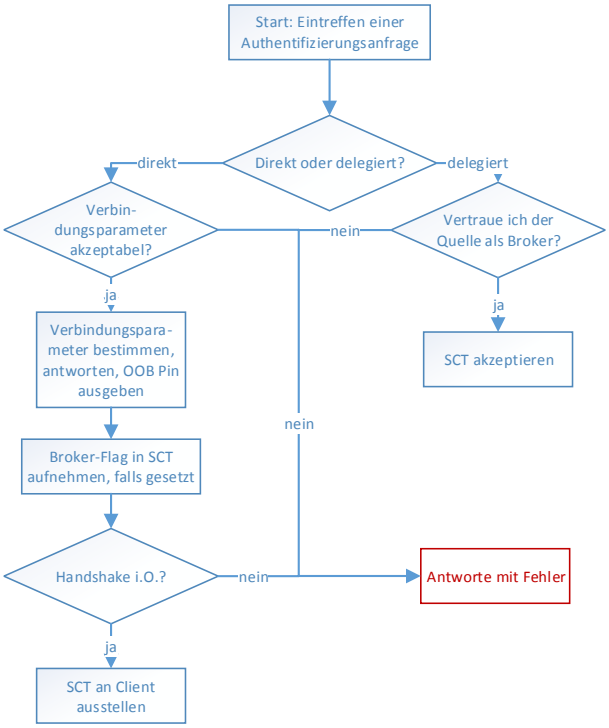


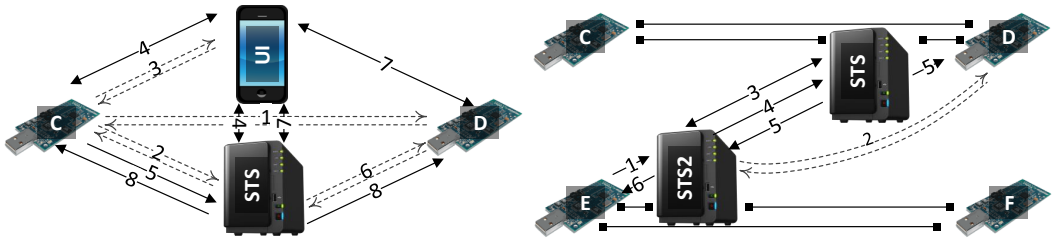
Abbildung 6.14: Aufgaben des Authentifizierungsendpunkts während einer Authentifizierung

den können. Außerdem ist ersichtlich, dass eine delegierte Authentifizierung weniger Aufwand für einen Endpunkt darstellt, als eine (in)direkte Authentifizierung, was in Abschnitt 6.4 detailliert diskutiert wird. Hinzu kommt die notwendige Nutzerinteraktion bei letzterem. Aus diesem Grund wurde die Authentifizierungsinfrastruktur von DPWSec so ausgelegt, dass einer delegierten Authentifizierung immer Vorzug gegeben wird.

Umsetzung verschiedener Authentifizierungsszenarien

Verhalten sich die einzelnen Geräteklassen wie beschrieben, so lassen sich damit verschiedenen Authentifizierungsszenarien realisieren. Diese werden im Folgenden unabhängig voneinander beschrieben, um die Verständlichkeit zu erhöhen, sie schließen sich jedoch nicht gegenseitig aus. Vielmehr sind sie alle in der selben Umgebung zeitgleich durchführbar.

Das Basisszenario ist in Abbildung 6.15 dargestellt. Zunächst werden im Netzwerk der Client „*C*“ und das Device „*D*“ angenommen. Der Client möchte eine Vertrauensbeziehung zum Device aufbauen und nutzt dazu die Hilfe



(a) Formung eines neuen Netzwerks (b) Beitritt weiterer Geräte

Abbildung 6.15: Darstellung des Basisszenarios

des Authenticators *„STS“* und des UI-Authenticators *„UI“*. In Schritt 1 erfährt der Client, dass das Device – ebenso wie er selbst – noch keinen Broker besitzt. Daher sucht es zunächst einen beliebigen Broker im Netzwerk (2) und anschließend einen UI-Authenticator (3) um sich darüber mit STS indirekt zu authentifizieren (4). Anschließend fordert der Client in Schritt 5 von dem nun ihm zugewiesenen Broker eine Authentifizierung mit D an. Der Authenticator überprüft die Broker von D (6) und beschließt, sich dem Device selbst als Broker anzubieten. Dies geschieht während der indirekten Authentifizierung über den UI-Authenticator (7). Durch die nun entstandene Vertrauensbeziehung zwischen STS und D kann STS aufgrund der ursprünglichen Anfrage (5) einen Sicherheitskontext für die Beziehung zwischen Client und Device erstellen und im letzten Schritt (8) an beide ausliefern. Die Authentifizierung ist abgeschlossen.

Dieses Szenario wird nun um weitere Teilnehmer erweitert. Die entstandenen Vertrauensbeziehungen sind im oberen Teil von Abbildung 6.15b zu sehen. Die im unteren Teil dargestellte Konstellation ist ebenso entstanden wie die obere. Möchte Client *„E“* nun mit Device D kommunizieren, so fordert er seinen Broker *„STS2“* zu einer delegierten Authentifizierung auf (1). STS2 überprüft die Broker von D (2) und stellt fest, dass eine Lücke in den Vertrauensbeziehungen durch eine (in)direkte Authentifizierung geschlossen werden muss (3), da die beiden Broker sich noch nicht vertrauen. Ist die Vertrauensbeziehung zwischen den beiden Authenticator-Geräten STS und STS2 aufgebaut, kann STS2 den ursprünglichen Request 1 in Schritt 4 weiterleiten. STS generiert ein SCT und liefert es in Schritt 5 sowohl an das Target als auch an STS2 aus. Letzterer kann somit in Schritt 6 die ursprüngliche Anfrage des Clients beantworten und die Vertrauensbeziehung zwischen E und D ist aufgebaut. Nicht eingezeich-

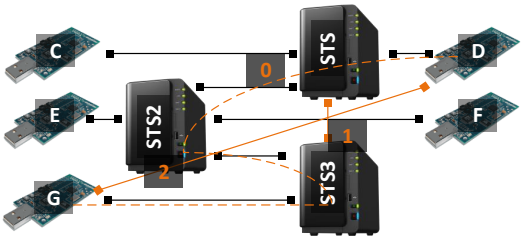
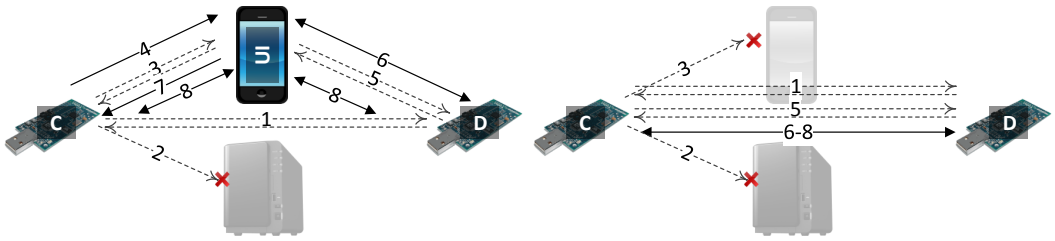


Abbildung 6.16: Transparente Auflösung langer Trust-Pfade

net ist der Fall, dass anschließend der Client C mit Device F kommunizieren möchte. In diesem Fall ist keine Nutzerinteraktion in Form einer (in)direkten Authentifizierung mehr notwendig, da die Vertrauensbeziehung zwischen STS und STS2 bereits besteht und die Delegation der Vertrauensbeziehung für den Nutzer unbemerkt im Hintergrund abläuft.

Im weiteren Verlauf des Aufbaus eines Netzwerks wird die Notwendigkeit für eine (in)direkte Authentifizierung und dem damit einhergehenden kryptografischen Handshake und der benötigten Nutzerinteraktion abnehmen. Abbildung 6.16 zeigt die Erweiterung des Szenarios aus Abbildung 6.15b. Die bestehenden Vertrauensbeziehungen sind schwarz gekennzeichnet. In diesem Szenario wendet sich Client G an seinen Broker STS3 mit der Bitte, eine delegierte Authentifizierung mit Device D aufzubauen. Es existiert bereits ein indirekter Trust-Pfad (0), wodurch keine Nutzerinteraktion notwendig ist. Zur Auflösung des Pfads wird STS3 wie gewohnt D nach seinen Broker fragen und daraufhin die Authentifizierung mit STS beginnen. Diese lässt sich delegiert bewerkstelligen, wodurch transparent eine neue Vertrauensbeziehung zwischen STS3 und STS entsteht (1). STS generiert anschließend den Sicherheitskontext für G und D und liefert diese an das Device sowie an STS, der ihn an den Client G weiterleitet (2).

Durch dieses Verfahren kann im Verlauf der Zeit ein Netzwerk an Authenticator-Geräten entstehen. Fällt eines von ihnen aus, können die anderen Geräte seine Aufgaben übernehmen, wobei die notwendige Nutzerinteraktion zur Wiederherstellung verlorengegangener Vertrauensbeziehungen gering gehalten wird. So wird die Einführung eines Single-Point-of-Failure vermieden. Eine formulierte Anforderung war jedoch, dass starke Teilnehmer die sichere Vernetzung von Kleinstgeräten zwar unterstützen sollen, ihr vollständiges Fehlen jedoch nicht zum Versagen des Netzwerks führen darf. Daher wurden die Ab-



(a) Authentifizierung ohne Authenticator (b) Authentifizierung ganz ohne Hilfe

Abbildung 6.17: Ablauf der Authentifizierung ohne Zuhilfenahme starker Teilnehmer

läufe und Schnittstellen so ausgelegt, dass die Kleinstgeräte Vertrauensbeziehungen auch mit weniger oder ganz ohne Hilfe aufbauen können. Fehlt zum Beispiel wie in Abbildung 6.17a ein Authenticator, so wird die Suche danach fehlschlagen (2). Der Client sucht daher einen UI-Authenticator (3), um eine indirekte Authentifizierung mit dem Device zu erbitten (4). Letzterer fragt die zur Verfügung stehenden Authentifizierungsmechanismen vom Device an (5) und führt anschließend die indirekte Authentifizierung für C mit D durch (6-8). Fehlt im Netzwerk wie in Abbildung 6.17b auch ein (passender) UI-Authentifizierer (3), so muss der Client versuchen, eine direkte Verbindung zum Device herzustellen. C fragt daher die unterstützten OOB-Mechanismen und Verbindungsparameter an (5) und legt eine Authentifizierungsmethode sowie Parameter wie Verschlüsselungsalgorithmen und Schlüssellängen für den aufzubauenden Sicherheitskontext fest. Anschließend führt der Client eine direkte Authentifizierung mit dem Device durch (6-8). Für das Device bedeuten beide Alternativen in Abbildung 6.17 den selben Aufwand. Der Client hat hingegen bei der direkten Authentifizierung erhöhte Aufwände insbesondere bei der Festlegung der Verbindungsparameter. Deutlicher werden die Unterschiede bei einer größeren Anzahl aufzubauender Vertrauensbeziehungen. Bei der indirekten Authentifizierung kann der Client den selben UI-Authenticator wiederverwenden, wodurch seine Aufwände bei den Discovery- und Description-Vorgängen reduziert werden. Ist ein Authenticator vorhanden, sind sowohl der komplexe kryptografische Handshake als auch die damit einhergehende Nutzerinteraktion nur einmal je ressourcenbeschränktem Gerät notwendig. Eine detaillierte Betrachtung erfolgt in Abschnitt 6.4.

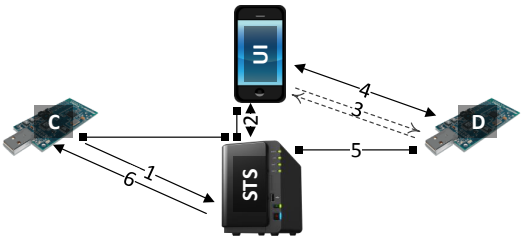


Abbildung 6.18: Ausnutzung einer Vertrauensbeziehung zwischen UI-Authentifizierer und Authentifizierer

Ein weiteres Szenario, dass sich durch die beschriebenen Teillösungen umsetzen lässt, betrifft die Benutzbarkeit der Authentifizierungsinfrastruktur. Multimediageräte der Klasse UI-Authenticator wurden bisher lediglich als transparente Übersetzer von OOB-Kanälen betrachtet. Integriert man jedoch einzelne dieser Geräte stärker in die Infrastruktur, so können sie gezielt Vertrauensbeziehungen untereinander und mit anderen Geräten wie zum Beispiel den Geräten der Authenticator-Klasse aufbauen. Letzteres sind in den anvisierten Szenarien z.B. NAS-Systeme, DSL-Modem/Router-Kombinationen, Kommunikationsanlagen und ähnliche Geräte. Sie befinden sich oft nicht in unmittelbarer Reichweite des Nutzers. Erlaubt man Multimediageräten, Vertrauensbeziehungen mit Authenticator-Geräten aufzubauen, kann bei künftigen Authentifizierungsvorgängen der Encrypted-Pin-Exchange eingesetzt werden, der in Kapitel 4.5.2 beschrieben wird. Der Vorgang wird in Abbildung 6.18 verdeutlicht. Vorausgesetzt werden je eine Vertrauensbeziehung zwischen Client C und dem Authenticator STS sowie zwischen STS und dem UI-Authenticator UI. Aufgrund der Anfrage des Clients zur delegierten Authentifizierung mit dem Device (1), initiiert STS eine (in)direkte Authentifizierung mit D. Durch Nutzung des in Abschnitt 6.3.2 beschriebenen Low-Effort-Discovery kann der Authenticator gezielt nach Multimediageräten suchen, mit denen er eine Vertrauensbeziehung unterhält. Das erlaubt, während einer indirekten Authentifizierung die OOB auszutauschende PIN zwischen UI und STS verschlüsselt in-band zu übertragen (2). Das halbiert mindestens die notwendige Nutzerinteraktion, da keine OOB-Pin an den unter Umständen für den Nutzer schwer zu erreichenden Authenticator übertragen werden muss.

Aufhebung von Vertrauensverhältnissen

Um bestehende Vertrauensbeziehungen aufzulösen, bietet der Authentifizierungsdienst eines Endpunktes das Cancel-Binding an, wie es in WS Trust definiert ist. Soll ein Gerät das Netzwerk verlassen, lassen sich so einzelne Sicherheitskontexte gezielt auflösen. Geräte der Authenticator-Klasse bieten außerdem eine Schnittstelle für einen *Federated Sign-out*, die in WS Federation festgeschrieben ist. Dadurch kann ein Client seinem primären Broker die Benachrichtigung zum dauerhaften Austritt aus dem Netzwerk geben, und dieser diese Benachrichtigung an alle Endpunkte verteilen.

Der Federated-Sign-out-Mechanismus ermöglicht es auch, dass ein Gerät bei Verlust oder Diebstahl über eine Management-Schnittstelle der Broker aus dem Netzwerk ausgetragen und alle Vertrauensbeziehungen dieses Geräts aufgelöst werden. Da die Broker im Besitz der zugehörigen Sitzungsschlüssel sind, können sie die Cancel-Operationen im Namen der auszutragenden Clients aufrufen. Um Clients über das Ausscheiden von Devices in Kenntnis zu setzen, definiert WS Federation einen Event-Filter. Clients müssen demnach bei der Verbindung mit ihrem primären Broker dessen Event-Quelle zur Auflösung von Vertrauensverhältnissen abonnieren.

Bei einer kaskadierten, delegierten Authentifizierung (vgl. Client C und Device F in Abbildung 6.15b) kann bei Austritt von Client C dessen primärer Broker STS das Cancel-Binding des Devices F aufrufen, da STS durch die Kaskadierung ebenfalls im Besitz des zugehörigen Sitzungsschlüssels ist. Soll hingegen F aus dem Netzwerk ausgetragen werden, muss $STS2$ einen Cancel-Request an STS stellen, der diesen anschließend über ein Event an den Client C ausliefert. Um die Funktion und Berechtigung dieser Vorgänge sicherzustellen, müssen Broker zu jeder delegierten Vertrauensbeziehung festhalten, ob und über welche weiteren Broker eine Kaskadierung stattgefunden hat.

6.3.6 Autorisierung

Das Schutzziel der Autorisierung unterscheidet sich grundlegend von den übrigen, da zu seiner Umsetzung keine kryptografischen Maßnahmen notwendig sind. Zu diesem Schluss kommen auch Swoboda et al. in [SSP08] und bezeichnen das Problem als „organisatorische Maßnahme“, wobei Kryptografie

nur zur „Authentifikation der Subjekte benutzt“ wird. Bei der Autorisierung wird auf Basis authentischer Informationen eine Entscheidung getroffen, ob ein Anfragesteller Zugriff auf die Ressource eines Objekts erhält oder nicht.

Klassische Systeme verwendeten Zugangskontrolllisten (Access Control Lists, ACL), die statische Konfigurationen enthalten und festschreiben, welche Nutzer wie auf welche Ressource zugreifen darf. Um die Konfiguration zu vereinfachen, wurden rollenbasierte Modelle eingesetzt (Role-based access control, RBAC). Dabei werden Nutzer Gruppen bzw. Rollen zugeordnet und den Rollen können wiederum Berechtigungen eingeräumt werden. RBAC-Modelle besitzen eine große Praxisrelevanz und werden in fast allen gängigen ERP-, Contentmanagement- oder Dokumentmanagementsystemen genutzt. Auch die heute gängigen Betriebssysteme basieren auf RBAC-Modellen.

Während ACL-basierte Zugriffskontrolle ein hohes Maß an manueller Konfiguration erfordert, sind RBAC-Modelle vor allem auf Nutzer und ihre Funktionen in einer organisatorischen Einheit zugeschnitten. Beides ist nicht ideal für intelligente Umgebungen, in denen vor allem Geräte miteinander kommunizieren sollen und der manuelle Konfigurationsaufwand gering sein soll.

Aktueller Forschungsgegenstand sind daher attributbasierte Zugriffskontrollmodelle (Attribute-based access control, ABAC) [HKF15]. Ihr Ziel liegt darin, Autorisierungen dynamisch auf Basis von (derzeitigen) Attributen des Anfragestellers, des Subjekts und der Umgebung zu generieren. ABAC-Modelle sind demnach ein vielversprechender Ansatz, kontextsensitive Autorisierungsmechanismen umzusetzen, die nicht den Nutzer, sondern die eigentlichen Ressourcen oder Geräte im Fokus haben.

Allen Autorisierungskonzepten ist gemein, dass sie nach dem Schema ablaufen, das in Abbildung 6.19 dargestellt ist. Ein Anfragesteller, der auf eine Ressource eines Subjekts zugreifen möchte, erbittet eine Autorisierung bei der dafür zuständigen Autorität und liefert dabei alle Informationen, die relevant sind, um eine qualifizierte Autorisierungsentscheidung zu treffen (1). In Schritt (2) muss ein nicht näher spezifizierter Prozess aus den ihm zur Verfügung stehenden Informationen eine Entscheidung generieren. Dieser Prozess kann das simple Abfragen einer ACL, das Einholen zusätzlicher Kontextinformationen oder die Delegation der Anfrage an eine qualifiziertere Autorität bedeuten. Ungeachtet der Komplexität dieses Prozesses resultiert er in Schritt

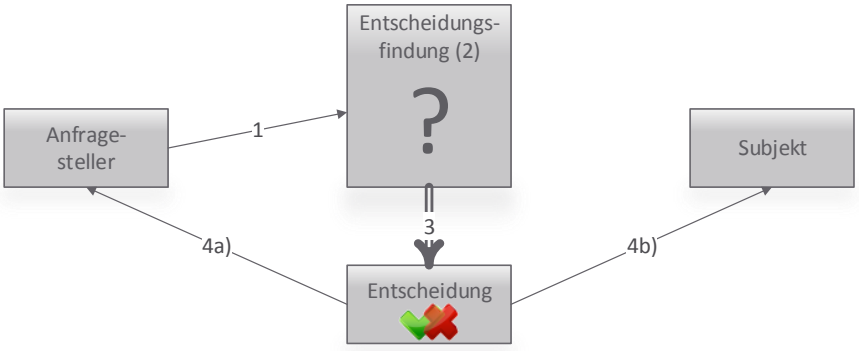


Abbildung 6.19: Ablauf einer Anfrage für eine Autorisierung

3 in einer Entscheidung zu der Frage, ob ein Anfragesteller unter den gegebenen Bedingungen auf die angefragte Ressource des Subjekts zugreifen darf. Abschließend muss diese Entscheidung kommuniziert werden (Schritt 4a/b).

Für die zu entwerfende Infrastruktur ist entscheidend, dass sie flexibel genug ist, um verschiedene Autorisierungskonzepte umzusetzen. In diesem Abschnitt werden zunächst die Teile der WS Security Spezifikationen beschrieben, die für die Anfrage, die Formulierung und die Auslieferung einer Autorisierung entscheidend und in DPWSec eingeflossen sind. Abschließend wird ein Autorisierungskonzept vorgestellt, das den Anforderungen aus Kapitel 5 genügt und auf der bereits bekannten Dreiteilung in starke, schwache und Multimediageräte beruht.

Autorisierung in DPWSec

Da die Herausforderung bei der Autorisierung die eigentliche Entscheidungsfindung betrifft, gestaltet sich auch die dazugehörige Kommunikation als vergleichsweise unkompliziert (vgl. Abb. 6.19). Es wird ein Mechanismus benötigt, um eine Autorisierung anzufordern (1). Dann muss ein nicht näher bestimmter Prozess auf Basis der ihm zur Verfügung stehenden Informationen eine Entscheidung treffen (2, 3), was keinen Einfluss auf die Gestaltung der Infrastruktur hat. Schließlich wird ein Mechanismus benötigt, der die Autorisierungsentscheidung ausliefert (4a/b). Im folgenden werden die Inhalte von Nachrichten und Dokumenten zur Anfrage und Auslieferung von Autorisierungen beschrieben. Die spezifizierten Nachrichtenformate sind in Anhang B.10 zu finden.

Zunächst muss das Gerät, für das eine Autorisierung erwirkt werden soll, in seinen Policies die Adresse des Security Token Services angeben, der berechtigt ist, Autorisierungen für das Subjekt auszustellen. Diese Aufgabe kann das Subjekt selbst oder eine dritte Partei übernehmen. Nun kann der Anfragesteller eine Autorisierung für das Subjekt anfordern (vgl. (1) in Abb. 6.19) und dabei alle Informationen angeben, die er für die Anforderung der Autorisierung für notwendig erachtet. Das Format einer solchen Anfrage an einen STS ist in WS Federation [L+06, Kpt. 9] festgelegt. Die Anfrage ähnelt der einer Authentifizierung, jedoch mit dem Unterschied, dass kein Sicherheitskontexttoken sondern ein SAML-Autorisierungstoken angefordert wird. Daneben enthält die Anfrage Behauptungen über den Anfragesteller – sogenannte Claims – und beliebige zusätzliche Kontextinformationen.

Claims dienen in erster Linie einer näheren Beschreibung des Anfragestellers, können jedoch auch jeden anderen Teilnehmer betreffen. Ein Claim kann aus einem Namen, einer Beschreibung, einem Wert und seiner darstellbaren Repräsentation bestehen. Claims stellen die Grundlage für die Entscheidungsfindung dar und sollten beispielsweise die Identifikation des Anfragestellers oder berechtigende Tokens wie Passwörter enthalten.

Schließlich kann eine Autorisierungsanfrage beliebige Kontextinformationen enthalten, die die Autorisierungsentscheidung beeinflussen können, jedoch nicht müssen. Eine solche Kontextinformation besteht aus einem Name-Wert-Paar und ist nicht weiter eingeschränkt. Ein denkbares Beispiel für eine Kontextinformation besteht darin, einen Wertebereich einzuschränken. So ließe sich ein Teilnehmer autorisieren, die Zieltemperatur eines Heizungsthermostats zwischen 20°C und 25°C zu wählen, während ein weiterer Teilnehmer die vollständige Kontrolle über den gesamten Wertebereich erhält.

Auf Basis der übermittelten Informationen muss nun die zuständige Autorität zu einer Entscheidung gelangen (vgl. 2 in Abb. 6.19). Das Format des entstehenden Dokuments, das die Entscheidung beinhaltet (vgl. 3 in Abb. 6.19), ist in der SAML-Core-Spezifikation ([CKPM05]) festgehalten. Es handelt sich hierbei um eine SAML-Assertion, die eine Autorisierungsentscheidung enthält.

Die SAML-Assertion beinhaltet zwingend die verwendete Version, eine Identifikation und eine Zeitangabe der Ausstellung des Dokuments sowie die Identifikationen der Autorität, die dieses Dokument ausgestellt hat und des Subjekts.

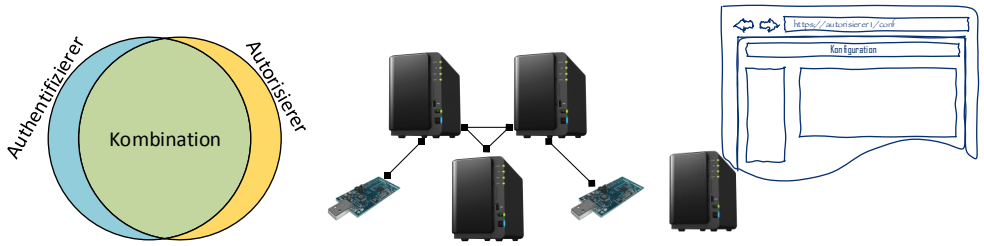
Weiterhin kann eine SAML-Assertion Bedingungen enthalten, unter denen die Assertion gültig ist und eine Signatur, die Integrität und Authentizität des Dokuments garantiert. Schließlich ist die eigentliche Autorisierungsentscheidung enthalten. Diese enthält die Ressource, für die die Autorisierung ausgestellt wurde, die tatsächliche Entscheidung und die erlaubten Aktionen, wie z.B. lesen oder schreiben. Optional können Beweise in Form weiterer SAML-Assertions enthalten sein, die zur Bildung der Entscheidung herangezogen wurden.

Die Beantragung und Auslieferung einer Autorisierung nach WS Federation und WS Trust und das SAML-Assertion-Format, das den Aufbau einer Autorisierungsentscheidung festlegt, sind die einzigen Teile der klassischen Spezifikationssuite, die Autorisierungen betreffen. Sie werden demnach vollständig und nahezu unverändert übernommen, um die Unterstützung verschiedener Autorisierungskonzepte nicht einzuschränken. Die einzige Änderung liegt in der Nutzung kompakter WSDD Compact Signatures anstelle der klassischen XML-Signaturen (vgl. Kapitel 3.2.1).

Autorisierung für intelligente Umgebungen

Die Entwurfsentscheidungen bei der Umsetzung eines Autorisierungskonzepts für intelligente Umgebungen ähneln denen beim Entwurf des Authentifizierungssystems aus Abschnitt 6.3.5. So sieht das Kommunikationsmodell von Web Services vor, dass ein Anfragesteller ein Autorisierungstoken in Form einer SAML-Assertion vom zuständigen STS erhält und diese bei jedem Request an das Subjekt mitliefert. Das Subjekt ist anschließend zuständig für die Validierung. Die Spezifikation WS Federation schlägt in [L+06, Kapitel 8.1] einen Referenzmechanismus vor. Dies verringert das notwendige Datenaufkommen, jedoch mit der Folge, dass das Subjekt für die Validierung mit dem STS in Verbindung treten muss, was das Kommunikationsmodell von DPWS verletzt. Aus diesem Grund wird das Validate-Binding, das für die delegierte Authentifizierung genutzt wird, ebenfalls zur Einspielung von Autorisierungsinformation verwendet.

Einem der Grundgedanken von DPWSec folgend, basiert das entwickelte Autorisierungskonzept auf der Dreiteilung der Geräteklassen. Neben den Geräten der Tiny-Klasse werden *synchrone Autorisierer* auf Geräten der Giant-



(a) Gerätekombinationen (b) Hohe Trust-Dichte (c) Wartungsschnittstellen

Abbildung 6.20: Getroffene Annahmen für das vorgestellte Autorisierungskonzept

Klasse gehostet. Multimediageräte werden aufgrund ihrer Nähe zum Nutzer und ihrer vielseitigen Schnittstellen als *asynchrone Autorisierer* eingesetzt. Die Begriffe synchron und asynchron beschreiben den Zeitpunkt, zu dem der Nutzer eine Autorisierung festlegt. Ein synchroner Autorisierer verfügt über eine Konfigurationsoberfläche, über die der Nutzer Clients für die Nutzung einzelner Ressourcen autorisieren kann. Möchte ein Client auf eine Ressource eines Subjekts zugreifen, ohne dass auf synchronem Weg eine Autorisierung festgelegt wurde, wird im Moment des Zugriffs eine Nutzerentscheidung über einen asynchronen Autorisierer eingeholt.

Für das hier vorgestellte Autorisierungskonzept werden die folgenden Annahmen getroffen, die in Abbildung 6.20 dargestellt sind. Zunächst wird angenommen, dass die vorhandenen Instanzen der Authenticator-Geräte und die vorhandenen Instanzen der synchronen Autorisierer weitestgehend paarweise auf den selben Geräten gehostet werden (vgl. Abb. 6.20a). Somit kann sich ein Gerät, das sich während der Authentifizierung als Broker anbietet ebenfalls als primärer synchroner Autorisierer für ein Tiny-Gerät anbieten. Weiterhin wird angenommen, dass auf Basis des in Abschnitt 6.3.5 beschriebenen Authentifizierungskonzepts eine hinreichende Anzahl an Vertrauensbeziehungen besteht (vgl. Abb 6.20b), sodass die Tiny Geräte den Autorisierern vertrauen oder Vertrauensverhältnisse möglichst ohne Nutzerinteraktion ad hoc aufgebaut werden können. Schließlich wird angenommen, dass synchrone Autorisierer eine Konfigurationsoberfläche bereitstellen, über die Autorisierungen synchron erteilt und Nutzer-Credentials hinterlegt werden können (vgl. Abb 6.20c).

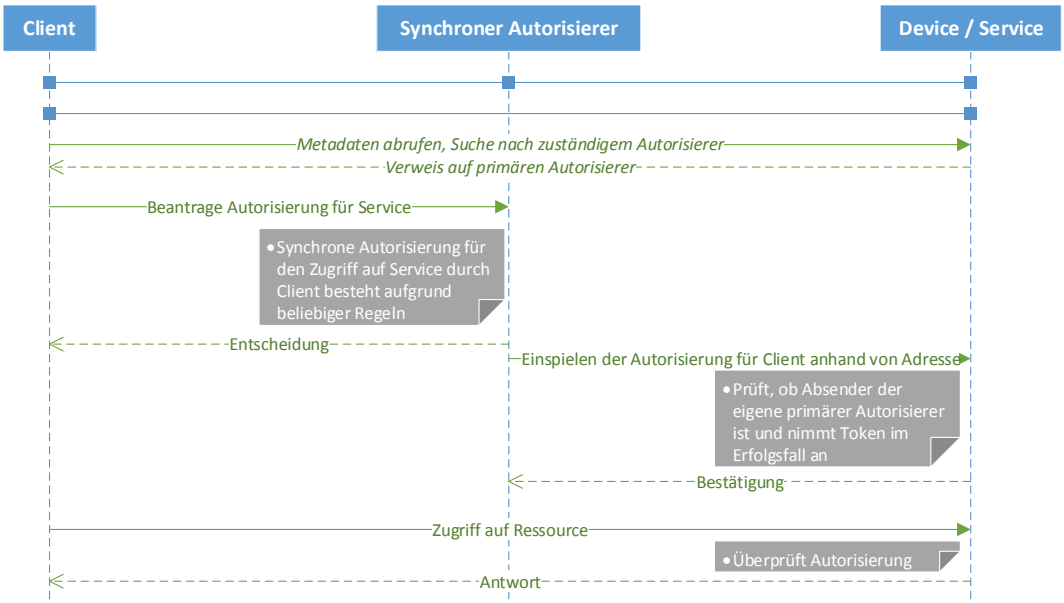


Abbildung 6.21: Szenario der synchronen Autorisierung

Autorisierungsszenarien Im Folgenden werden verschiedene Szenarien diskutiert. Abbildung 6.21 zeigt den einfachsten Fall, in dem eine statische Konfiguration durch den Nutzer vorliegt, die den Client ermächtigt, auf eine Ressource eines Dienstes zuzugreifen. Aus den Metadaten des Dienstes, den der Client ansprechen möchte, erfährt er die Adresse des synchronen Autorisierers, der für das Subjekt zuständig ist. Der Client beantragt beim primären Autorisierer des Subjekts eine Autorisierung unter Angabe aller notwendigen Claims wie z.B. seine Adresse, sein Standort oder sein Typ. Da dem Autorisierer auf synchronem Weg durch den Nutzer eine Regel gegeben wurde, die es dem Client aufgrund seiner Claims erlaubt, auf den Dienst und seine Ressource zuzugreifen, erteilt er dem Client die Erlaubnis. Außerdem spielt er dem Subjekt über seine Validate-Schnittstelle eine Autorisierungsregel auf Basis der Identifikation des Clients ein, sodass die Autorisierungsprüfung bei künftigen Aufrufen zugunsten des Clients ausfallen. Es sei erwähnt, dass der primäre synchrone Autorisierer für das Gerät durch das Gerät selbst gehostet werden kann. Somit ist es einem Gerät möglich, seine Autorisierungen selbst zu verwalten, falls es auf die Unterstützung von stärkeren Teilnehmer verzichten möchte.

Abbildung 6.22 zeigt, wie die Situation gelöst werden kann, wenn der zuständige synchrone Autorisierer keine Autorisierungsentscheidung treffen kann.

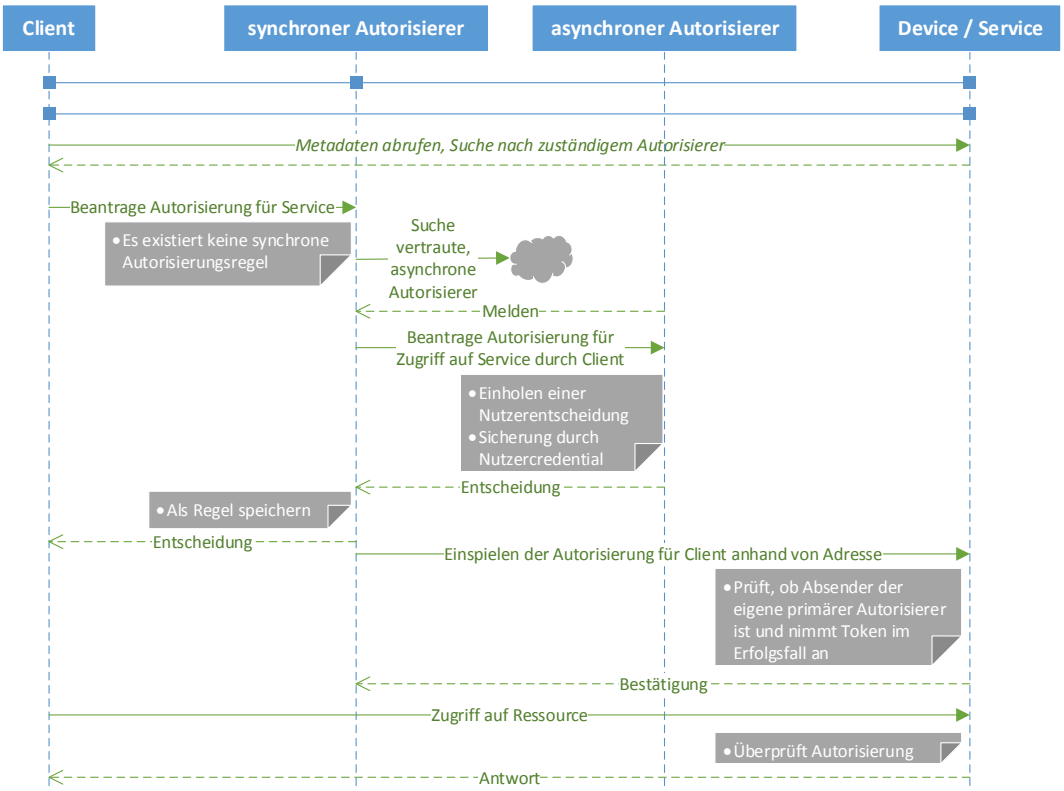


Abbildung 6.22: Szenario der asynchronen Autorisierung

Durch das Ansprechen eines vertrauten asynchronen Autorisierers – zum Beispiel eines Smartphones – wird die Beschreibung aus den Claims des Anfragestellers einem Nutzer präsentiert, der anschließend eine Entscheidung trifft. Um sicherzustellen, dass die Entscheidung im Sinne des Nutzers geschieht, müssen synchrone und asynchrone Autorisierer über mindestens ein gemeinsames Nutzer-Credential verfügen. Es kann sich dabei um ein Passwort oder ein biometrisches Merkmal handeln und wird über die Konfiguration der synchronen Autorisierer festgelegt bzw. während einer asynchronen Autorisierung vom Nutzer erfragt und gespeichert. Ist der angesprochene asynchrone Autorisierer z.B. aufgrund mangelnder Nutzer-Credentials nicht in der Lage, eine Entscheidung zu treffen, wird der nächste vertraute asynchrone Autorisierer angesprochen, bis eine Entscheidung getroffen werden kann. Im Falle eines Misserfolgs wird die Suche nach asynchronen Autorisierern mit einem High-Effort-Discovery ausgeweitet. Die Bevorzugung vertrauter asynchroner Autorisierer hat keine kryptografischen Gründe, sondern geschieht, da die Wahr-

scheinlichkeit höher ist, dass sie die Berechtigung durch den Nutzer erhalten haben.

Der letzte umgesetzte Mechanismus dient der Verbesserung der Benutzbarkeit und der Erhöhung der Ausfallsicherheit. In den zuvor dargestellten Szenarien in den Abbildungen 6.21 und 6.22 können keine Autorisierungen für ein Subjekt mehr eingeholt werden, wenn dessen primärer Autorisierer nicht verfügbar ist. Er bietet damit ein Ziel für gezielte Denial-of-Service-Angriffe (DoS), da mit seinem Ausscheiden auch alle Geräte, für die er zuständig ist, nur noch eingeschränkt verfügbar sind. Ein weiteres Problem besteht darin, dass der Nutzer den Überblick über die Zuordnung von Subjekten und deren primären Autorisierern behalten muss, um über die Konfigurationsschnittstelle weitere statische Autorisierungen hinzuzufügen.

Das Szenario in Abbildung 6.23 löst diese Probleme, indem es das Konzept des sekundären Autorisierers einführt. Es handelt sich dabei um synchrone Autorisierer, die die Aufgaben des primären Autorisierers eines Subjekts bei einem Ausfall übernehmen können und ermöglichen eine dezentrale synchrone Autorisierungskonfiguration. Im abgebildeten Szenario existieren der Client C und das Device D. Der primäre Autorisierer des Devices ist A_1 (1). Weiterhin existiert im Netzwerk ein weiterer synchroner Autorisierer A_2, über dessen Konfigurationsschnittstelle ein Nutzer unter Angabe eines passenden Credentials dem Client C Zugriffsrechte auf das Device D einräumt. Da A_2 noch nicht als sekundärer Autorisierer von D konfiguriert ist, nimmt er die stattgefundene Konfiguration zum Anlass, dies nachzuholen.

Hierzu bringt er in Erfahrung, dass A_1 der primäre Autorisierer des Subjekts D ist (3) und erbittet Autorisierung für eine spezielle Ressource des Subjekts, was seine Absicht erklärt, sich als sekundärer Autorisierer anzubieten (4). Durch Angabe eines passenden Nutzer-Credentials kann A_2 nachweisen, dass er dabei im Sinne des Nutzers handelt. Über seine validate-Schnittstelle wird dem Device D unter Angabe der bereits genannten speziellen Ressource mitgeteilt, dass sich A_2 als sekundärer Autorisierer anbietet. Akzeptiert das Device (6), wird das auch A_2 mitgeteilt (7) und seine Rolle als sekundärer Autorisierer ist bestätigt (8).

Will der Client nun das Device ansprechen und eine Autorisierung einholen, so verläuft dies zunächst wie im synchronen Basisszenario (9,10). Bevor jedoch

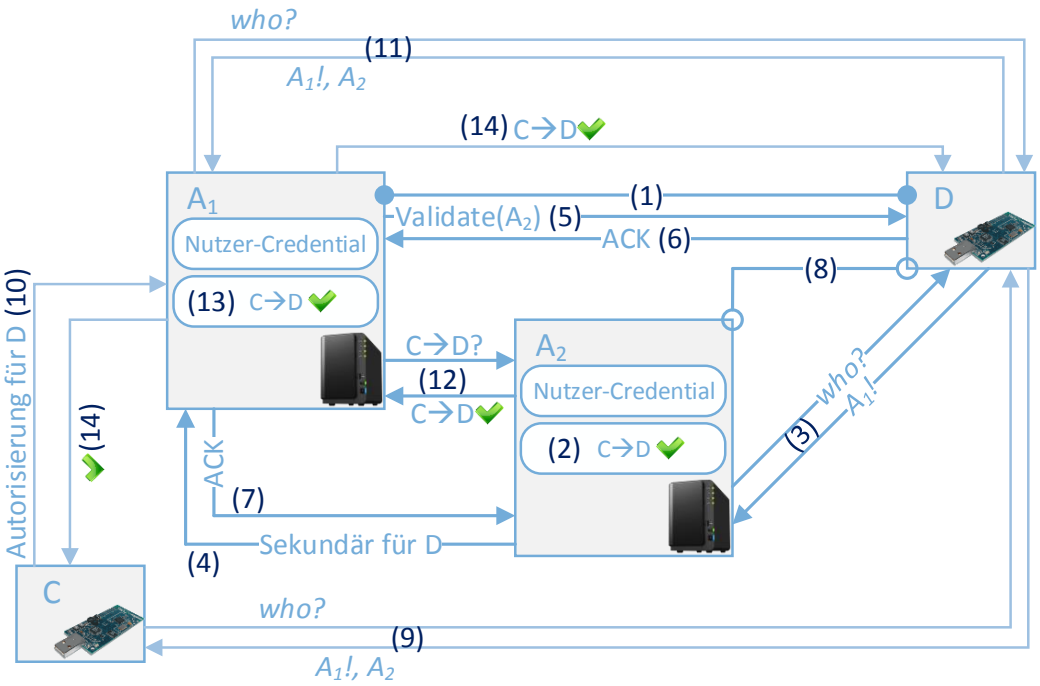


Abbildung 6.23: Erhöhung der Ausfallsicherheit und Nutzerfreundlichkeit durch das Konzept der sekundären Autorisierer

der primäre Autorisierer des Subjekts einen asynchronen Autorisierer zu Rate zieht, holt er zunächst Informationen zu den sekundären Autorisierern des Subjekts ein (11) und prüft, ob so eine Autorisierungsentscheidung eingeholt werden kann (12). Um künftige Abfragen zu vermeiden und die Redundanz zu erhöhen, speichert er das Ergebnis zwischen (13) und liefert die Entscheidung an die betreffenden Teilnehmer aus (14).

Für den Fall, dass der primäre Autorisierer des Subjekts nicht verfügbar ist, spricht der Client den ersten sekundären Autorisierer des Subjekts an. Dieser fällt temporär in die Rolle eines primären Autorisierers, befragt die übrigen sekundären Autorisierer und holt bei Misserfolg eine Nutzerentscheidung über einen asynchronen Autorisierer ein. Schließlich sei erwähnt, dass ein Subjekt auch sein eigener sekundärer Autorisierer ist. In diesem Fall handelt es sich um eine Fallback-Lösung, falls die übrigen unterstützenden Geräte nicht zur Verfügung stehen.

Aufgaben der Geräteklassen Zur Umsetzung der beschriebenen Szenarien haben die verschiedenen Geräteklassen die folgenden Aufgaben zu erledigen.

Ein Device der Tiny-Klasse bietet die Validate-Schnittstelle an, über die neue Autorisierungen eingespielt und weitere sekundäre Autorisierer bestimmt werden können. Letztere muss das Device in den Policies seines Sicherheitsdienstes publizieren. Das Device muss sicherstellen, dass die Validate-Schnittstelle ausschließlich von vertrauten Geräten angesprochen wird, die zuvor als Autorisierer registriert wurden. Ein Client, der einen Dienst ansprechen möchte, fragt die Beschreibung des Subjekts ab, um die Adressen seiner Autorisierer zu erfahren. Er gibt dabei dem Primären Vorzug. Nur wenn dieser nicht verfügbar ist, fällt der Client auf den ersten erreichbaren sekundären Autorisierer zurück. Die asynchronen Autorisierer hosten ebenfalls einen STS, über den Autorisierungstokens beantragt werden. Sie benötigen eine Nutzerschnittstelle, die die Claims aus der Anfrage übersichtlich darstellen und anschließend eine z.B. zeitlich einschränkbare Entscheidung treffen.

Die eigentliche Auflösung übernehmen die synchronen Autorisierer (vgl. Abbildung 6.24). Bei Eintreffen einer Autorisierungsanfrage muss zunächst geprüft werden, ob es sich um ein Angebot handelt, sekundärer Autorisierer für das Subjekt zu werden. Ist dies der Fall, wird das Credential geprüft und das Angebot wird weitergeleitet. Bestätigt das Subjekt die Annahme, wird dies dem neuen, sekundären Autorisierer quittiert, und der Vorgang ist beendet. Passt jedoch das angegebene Credential nicht oder akzeptiert das Subjekt keine (weiteren) Autorisierer, wird mit einem entsprechenden Fehler geantwortet.

Handelt es sich um eine reguläre Autorisierungsanfrage und ist der angesprochene Dienst der primäre Autorisierer des Subjekts, so prüft er zunächst seine eigene Konfiguration. Ist daraus keine Entscheidung abzuleiten, befragt er das Subjekt nach seinen sekundären Autorisierern und versucht, über sie eine Entscheidung einzuholen. Bleibt auch das ergebnislos, befragt er zunächst vertraute und schließlich alle vorhandenen asynchronen Autorisierer, um eine Nutzerentscheidung einzuholen. Sowie eine der Maßnahmen Erfolg hat, wird die Entscheidung an den Anfragesteller und das Subjekt ausgeliefert und der Vorgang ist beendet.

Ist der angesprochene Dienst jedoch nicht der primäre, sondern eine sekundärer Autorisierer des Subjekts, muss er unterscheiden, aus welchem Grund er angesprochen wurde. Stellt der Teilnehmer die Anfrage, für den eine Autorisierung erwirkt werden soll, so geschieht dies aus dem Grund, dass er den

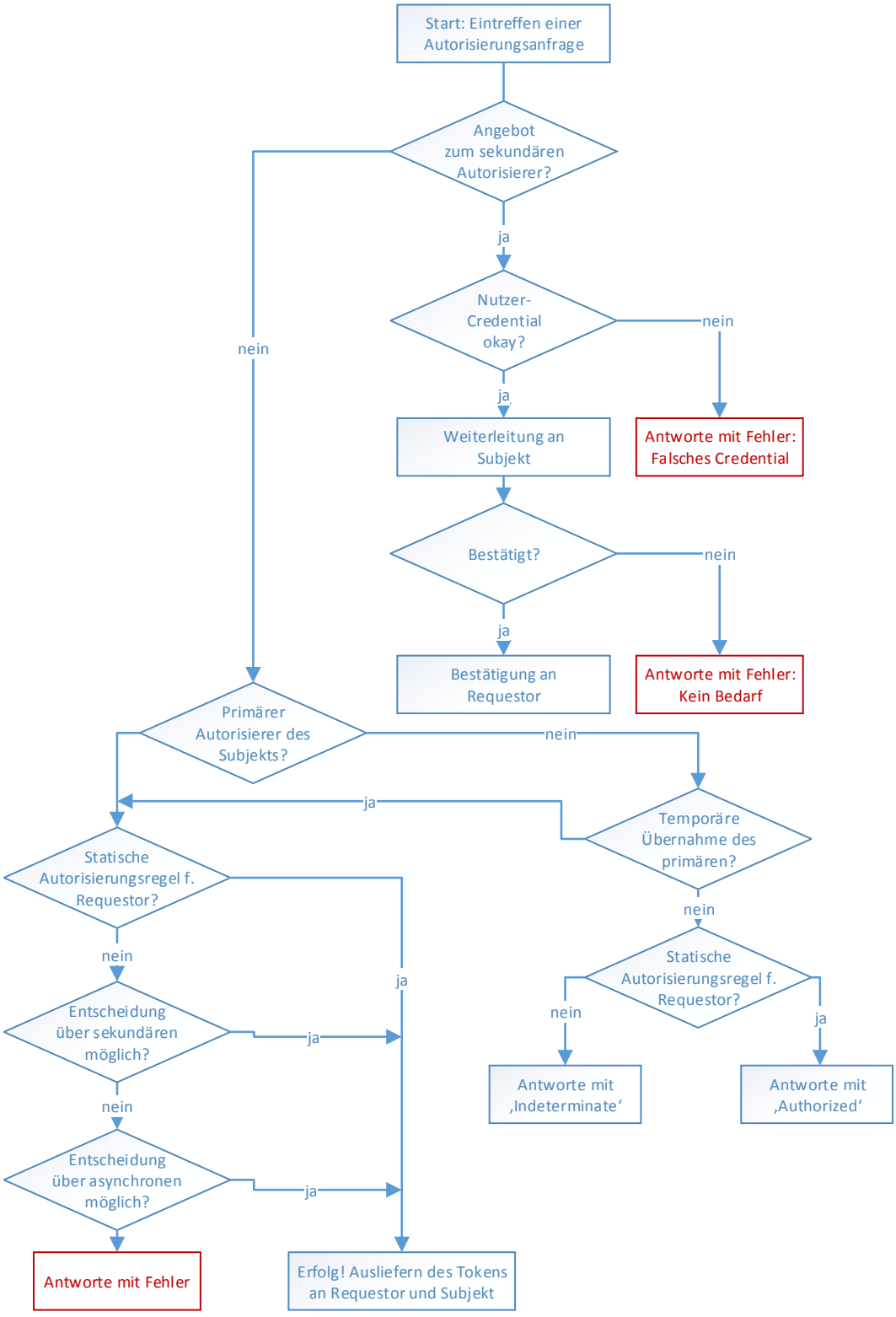


Abbildung 6.24: Interne Abläufe innerhalb eines synchronen Autorisierers

primären Autorisierer des Subjekts nicht erreichen konnte und der angesprochene Dienst nun temporär diese Rolle übernehmen muss. Andernfalls handelt es sich beim Anfragesteller um den primären Autorisierer des Subjekts. In diesem Fall antwortet der Dienst nur mit einer positiven Antwort oder gibt an, dass er keine Entscheidung treffen kann und versucht nicht, weiter synchrone oder asynchrone Autorisierer anzusprechen.

Für den speziellen Fall, dass ein ressourcenbeschränktes Gerät seine Autorisierungen selbst verwaltet und demnach sein eigener primärer Autorisierer ist, vereinfacht sich der beschriebene Ablauf. Sekundäre Autorisierer müssen nur zugelassen werden, falls die zentralisierte Konfiguration für den Nutzer vereinfacht werden soll. Die Fragestellung, ob der Dienst selbst der primäre Autorisierer für das Subjekt ist, stellt sich nicht und wird immer positiv beantwortet. Der Zweig, der die Aktionen für sekundäre Autorisierer enthält, entfällt demnach vollständig.

6.4 Prototypische Implementierung

Zum Nachweis seiner Funktionalität wurde das DPWSec prototypisch implementiert. Die in Kapitel 4.3.2 beschriebenen Hardware-Prototypen einer intelligenten Lampe und eines intelligenten Schalters wurden exemplarisch je als Device und als Client der Tiny-Klasse genutzt. Auch der WS4D Mobile Authenticator, der als Android-App die Nutzerinteraktion unterstützt, wurde wie beschrieben eingesetzt. In dieser Hinsicht gilt die in Abbildung 4.8 dargestellte Software-Architektur unverändert, und auch die angegebenen Hardware-Spezifikationen haben sich nicht geändert.

Hinzugekommen ist die Umsetzung eines Giant-Geräts, das die Funktionen der delegierten Authentifizierung und der synchronen Autorisierung bereitstellt, um die das JMEDS-Framework ergänzt wurde. Der WS4D Mobile Authenticator beherbergt zusätzlich die notwendigen Funktionen für die asynchrone Autorisierung. Die Implementierung des Client-Geräts wurde ebenfalls auf einem Raspberry Pi ausgeführt. Hierbei stand jedoch nicht die Eigenschaft des schnellen Prototypings im Vordergrund sondern die Tatsache, dass die Leistungsfähigkeit des Raspberry Pis der eines Authentifizierers in den Szenarien

aus Kapitel 5 entspricht, wie etwa Heim-NAS-Systemen oder Modem-Router-Kombinationen.

Die beschriebenen Konzepte wurden umgesetzt. Betritt der Lichtschalter das Netzwerk, ermittelt er die vorhandene Lampe als zu schaltendes Ziel, dessen Metadaten noch keinen Broker enthalten. Da der Schalter selbst ebenfalls noch keinen Broker bestimmt hat, sucht er den ersten verfügbaren, führt eine (in)direkte Authentifizierung mit ihm durch und beantragt anschließend eine delegierte Authentifizierung mit der Lampe bei seinem frisch ernannten Broker. Letzterer ermittelt, dass das Authentifizierungsziel noch keinen eigenen Broker besitzt und bietet sich daher als Broker an, während er die (in)direkte Authentifizierung mit der Lampe durchführt. Anschließend kann die angefragte delegierte Authentifizierung zwischen Schalter und Lampe ohne weitere Nutzerinteraktion durchgeführt werden. Da der Broker, wie im Szenario aus Abschnitt 6.3.6 gleichzeitig beschrieben, als synchroner Autorisierer agiert, bietet er der Lampe während der Authentifizierung an, als primärer Autorisierer zu fungieren. Daher kann der Schalter im Anschluss an die Authentifizierung mit der Lampe beim implementierten Giant-Device eine Autorisierung für den Zugriff auf die Lampe erbitten. Die Implementierung des Prototyps unterstützt sowohl das Hinterlegen synchroner Autorisierungsregeln, als auch das asynchrone Einholen von Nutzerentscheidungen über ein verfügbares Multimediagerät, das im Prototypen durch ein Android-Smartphone realisiert wird.

Das Giant-Device wurde ebenfalls in Java implementiert und die in Kapitel 4.3.3 beschriebene Implementierung des WS4D Mobile Authenticators wurde erweitert, um asynchrone Autorisierungen zu ermöglichen. In Tabelle 6.3 ist der finale Umfang der einzelnen Prototypenkomponenten zum Zeitpunkt der Fertigstellung dieser Arbeit dargestellt. Es zeigt sich deutlich, dass die Änderungen fast ausschließlich das JMEDS-Framework betreffen, da die Aktionen zum möglichst transparenten Aufbau von Vertrauensbeziehungen, dem Einholen von Autorisierungsentscheidungen oder zur Durchsetzung von Sicherheitsrichtlinien durch das Framework erledigt werden. Die Umfänge der einzelnen Anwendungen für Schalter und Lampe haben sich beinahe gar nicht geändert und resultieren lediglich in Änderungen im Aufruf einzelner Funktionen des Frameworks. Die Ergänzung des WS4D Mobile Authenticators um eine Komponente zum Erfragen asynchroner Autorisierungen ändert die Größe

Komponente	Lines of Code
Erweiterungen JMEDS	13752
Crypto-Wrapper	1225
Bit Key Tool	132
Anwendung Lampe	543
Anwendung Schalter	1296
Anwendung Giant-Device	1337
WS4D Mobile Authenticator App	7023
Test- & Debugging-Software	269
Gesamt	**25577**

Tabelle 6.3: Auflistung der Umfänge der Prototypkomponenten

des Quellcodes ebenfalls lediglich um ca. 10%, da die gesamte innere Geräte- und Dienstlogik sowie die Logik zur Verarbeitung asynchron eintreffender Ereignisse (vgl. Abb. 4.11, S.85) wiederverwendet wurden.

Auf Basis der prototypischen Implementierung konnten Laufzeitanalysen der entstehenden Kommunikation unternommen werden. Untersucht wurden die Lauf- und Ausführungszeiten bei einer indirekten Authentifizierung, einer delegierten Authentifizierung sowie den anschließenden Operationsaufrufen im angegebenen Szenario aus einem Lichtschalter, einer Lampe und einem Giant-Device, die jeweils auf einem Raspberry Pi ausgeführt wurden. Ergänzt wird das Setup um ein Samsung Nexus S, das die WS4D Mobile Authenticator App ausführt. Tabelle 6.4 und Abbildung 6.25 zeigen die ermittelten Ausführungszeiten und Round Trip Times bei einer indirekten Authentifizierung. Die Bearbeitung der kryptografischen Routinen nimmt in einem Device bei beiden Hälften des Handshakes ca. 5 Sekunden in Anspruch. Die RTT liegen jedoch über dem Doppelten, da sie Initialisierungs- und Ausführungszeiten von Device und Smartphone beinhalten (vgl. Formeln 6.1 und 6.2). Die gemessenen Zeiten wurden bereinigt, indem die Discovery- und Description-Vorgänge vor dem Handshake nicht berücksichtigt wurden. Außerdem wurde die Messung während Nutzerinteraktionen (Pin-Eingabe, etc. . .) angehalten.

$$RTT_1 = t_{exec-client} + t_{exec-phone} + t_{exec-device} + t_{init-phone} + t_{init-device} \quad (6.1)$$

$$RTT_2 = t_{exec-client} + t_{exec-phone} + t_{exec-device} \quad (6.2)$$

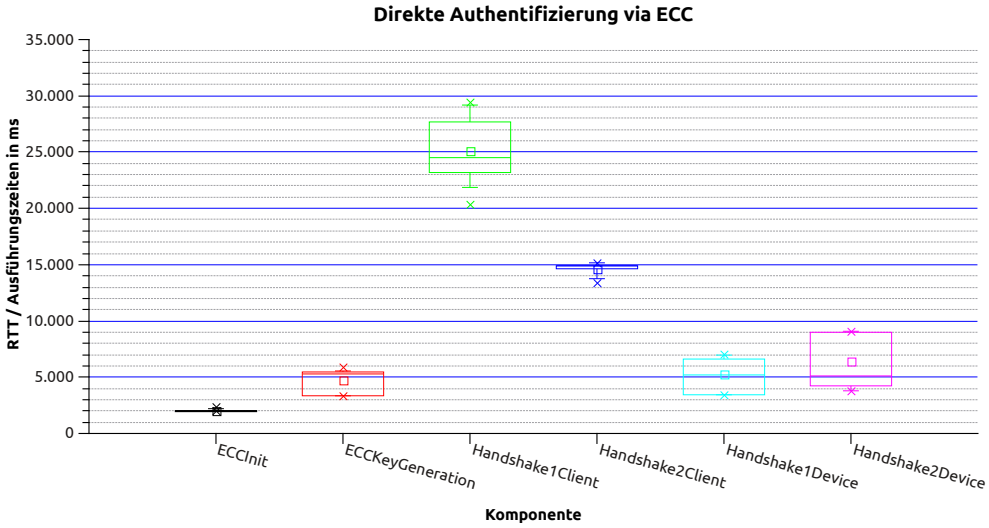


Abbildung 6.25: RTT und Ausführungszeiten bei indirekter Authentifizierung

Tabelle 6.5 und Abbildung 6.26 zeigen die RTT einer vollständigen, delegierten Authentifizierung sowie der Injektion eines Vertrauensverhältnisses in ein Device. Mit drei Sekunden liegt der Aufwand deutlich unter dem einer direkten Authentifizierung und beinhaltet zudem das Warten auf Discovery-Vorgänge im Authenticator, die sich durch Cache-Mechanismen abkürzen ließen. Außerdem läuft die delegierte Authentifizierung für den Nutzer völlig unbemerkt ab.

Mit dem während einer Authentifizierung ausgetauschten Schlüsselmaterial können die anschließenden Operationsaufrufe abgesichert werden. Tabelle 6.6 und Abbildung 6.27 zeigen die Round Trip Times von Operationsaufrufen,

Aufgabe	Zeit in ms
Handshake Teil 1 (RTT)	24.480
Handshake Teil 2 (RTT)	14.877
Handshake Teil 1 (Ausführung)	5.206
Handshake Teil 2 (Ausführung)	5.103
ECC Engine Initialisierung	2.024
ECC Schlüsselerzeugung	5.301

Tabelle 6.4: Ausführungszeiten bzw. RTT bei indirekter Authentifizierung (Mediane)

Aufgabe	RTT in ms
RTT Delegierte Authentifizierung	3.050
RTT Validate-Operation	687

Tabelle 6.5: RTT bei delegierter Authentifizierung (Median)

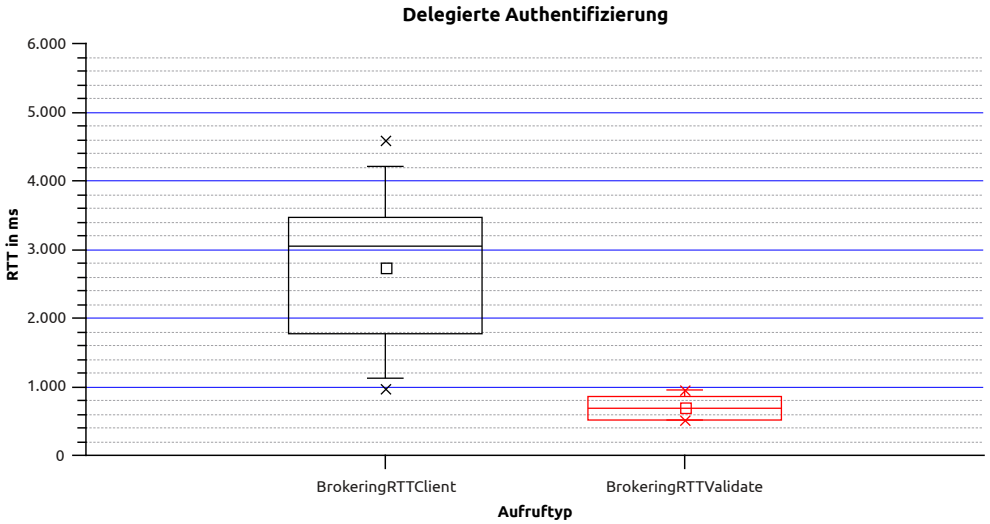


Abbildung 6.26: RTT bei delegierter Authentifizierung

die ohne Sicherheitsmechanismen, nur signiert oder signiert und verschlüsselt durchgeführt wurden. Die ausgetauschten Nachrichten, auf die sich diese Messwerte beziehen, sind in Anhang A.2 in den Listings A.13 bis A.18 dargestellt.

Art des Aufrufs	RTT in ms
Ohne Sicherheitsmaßnahmen	152
Nur Signaturen	156
Signiert und verschlüsselt	172

Tabelle 6.6: RTT gesicherter und ungesicherter Operationsaufrufe (Median)

Es ist ersichtlich, dass das Signieren einer Nachricht die RTT nur unwesentlich erhöht (ca. 2,6%). Das Verschlüsseln und Signieren einer Nachricht führt zu einer Erhöhung der RTT um ca. 13%.

Vergleicht man die Aufwände, die für eine indirekte und eine delegierte Authentifizierung betrieben werden müssen, wird ersichtlich, in welchem Maße starke Teilnehmer Aufwände von schwachen Teilnehmern verringern können.

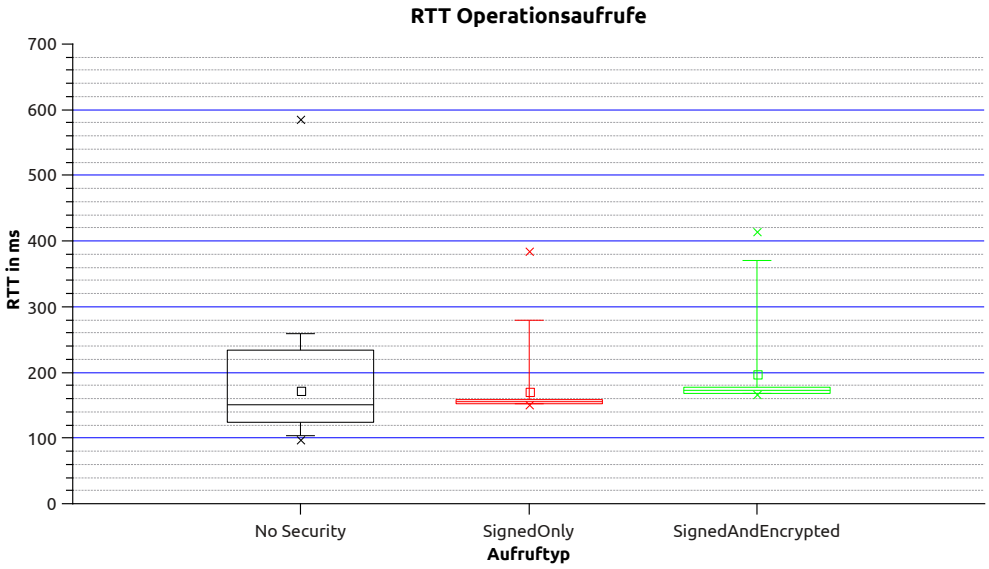


Abbildung 6.27: RTT gesicherter und ungesicherter Operationsaufrufe

Mit zunehmender Größe eines Netzwerks wird die Entlastung ersichtlicher, da für jedes zusätzliche Gerät der Aufwand einer indirekten Authentifizierung durch eine delegierte Authentifizierung ersetzt werden kann. Dennoch können bei Ausfall der Infrastruktur mit entsprechendem Mehraufwand Vertrauensbeziehungen zwischen Kleinstgeräten auch ohne Hilfe aufgebaut werden.

Stellt man die Aufwände von direkter und delegierter Authentifizierung in Abhängigkeit von den aufzubauenden Vertrauensbeziehungen gegenüber, ergeben sich die Zusammenhänge, die in Abbildung 6.28 dargestellt sind. Der Aufwand einer indirekten Authentifizierung wurde dabei auf den Wert 1 normiert. Mit steigender Anzahl zu unterhaltender Vertrauensbeziehungen steigen die Aufwände für beide Verfahren linear (vgl. Formeln 6.3 und 6.4), jedoch beträgt das Verhältnis des Aufwandes einer delegierten zu einer indirekten Authentifizierung den Faktor 0,065. Da der Delegation der ersten Vertrauensbeziehung eine (in)direkte Authentifizierung mit einem Authenticator-Device vorausgehen muss, ist in diesem Fall der Aufwand für die delegierte Authentifizierung höher. Mit jeder weiteren Vertrauensbeziehung sinkt der Gesamtaufwand je-

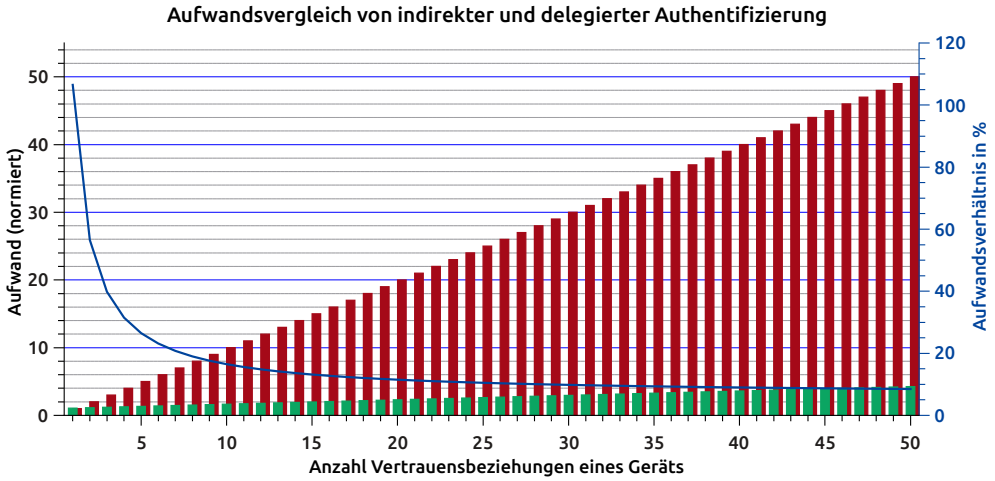


Abbildung 6.28: Gegenüberstellung der Aufwände von indirekter und delegierter Authentifizierung

doch. Für eine große Anzahl Vertrauensbeziehungen beträgt das Verhältnis lediglich 6,5% (vgl. Formeln 6.5 und 6.6).

$$Effort_{direct}(n) = n \tag{6.3}$$

$$Effort_{brokered}(n) = 1 + 0,065 \cdot n \tag{6.4}$$

$$ratio(n) = \frac{1 + 0,065 \cdot n}{n} \tag{6.5}$$

$$\lim_{n \to \infty} ratio(n) = 0,065 \tag{6.6}$$

6.5 Zusammenfassung und Diskussion

In diesem Kapitel wurden zunächst verschiedene Technologien zur Vernetzung von Kleinstgeräten vorgestellt und die in ihnen umgesetzten Sicherheitsmaßnahmen wurden beleuchtet. Es ließ sich zunächst erkennen, dass das Thema Kommunikationssicherheit insbesondere im Rahmen verwandter Forschungsarbeiten häufig nicht oder nur teilweise abgedeckt wird. Wurde Sicherheit betrachtet, basierte sie auf vollständigen Neuentwicklungen oder auf bestehenden Komponenten, die für die sichere Vernetzung von klassischen Rechnernetzen entworfen wurde. Keine der betrachteten Lösungen erfüllt alle Anforderungen, die in Kapitel 5 beschrieben und in Tabelle 5.2 zusammengefasst wurden.

Das vorgestellte DPWSec stellt eine infrastrukturelle Grundkomponente für die Vernetzung von Geräten in intelligenten Umgebungen dar. Auf eine Neuentwicklung von Grund auf wurde gezielt verzichtet. Stattdessen wurde mit der Web Service Security Spezifikationssuite auf eine bestehende Lösung zurückgegriffen. Aus ihr wurden mithilfe der entwickelten Methodik die Spezifikationsteile isoliert, die zur Vernetzung von Geräten heterogener Leistungsfähigkeit benötigt werden und so arrangiert, dass starke Teilnehmer eines Netzwerks die schwächeren entlasten können. Durch gezielte Berücksichtigung und Einbindung von nutzernahen Geräten ist Nutzerfreundlichkeit ein integraler Bestandteil von DPWSec.

Um eine Beurteilung von DPWSec vornehmen zu können, werden die in Kapitel 5.4 zusammengefassten Anforderungen herangezogen. Tabelle 6.7 zeigt eine Zusammenfassung.

Das zugrundegelegte Dolev-Yao-Angreifermodell wird vollständig berücksichtigt, da die Sicherungsmaßnahmen auf Nachrichtenebene einen unberechtigten Eingriff in die Kommunikation verhindern und durch die erarbeiteten Authentifizierungskonzepte MITM-Angriffen begegnet wird. Letzteres garantiert ebenfalls ein kontinuierliches, sicheres Deployment, da neue Geräte auf sichere Weise im laufenden Betrieb in ein bestehendes Netzwerk eingebracht werden können. Der herausgestellten Besonderheit intelligenter Umgebungen, dass Verlust oder Diebstahl eines einzelnen Geräts vergleichsweise wahrscheinlich ist, wird damit Rechnung getragen, dass ein Endpunkt ausschließlich ihn betreffendes Schlüsselmaterial enthält. Im Gegensatz z.B. zur Verwendung netzwerkweiter Schlüssel sind die Verbindungen der übrigen Geräte nicht betroffen. Außerdem wird ein Mechanismus umgesetzt, der das koordinierte Austragen eines Geräts aus dem Netzwerk bewirkt, wodurch alle bestehenden Vertrauensbeziehungen aufgelöst werden.

DPWSec berücksichtigt die Leistungsheterogenität der Teilnehmer, da schwache Geräte entlastet werden, indem stärkere Geräte Aufwände übernehmen. Gleichzeitig wird die Einführung von Single-Points-of-Failure vermieden, da der Wegfall einzelner Authentifizierer oder Autorisierer kompensiert werden kann. Dieses Verhalten reicht so weit, dass schwache Geräte die Unterstützung stärkerer Geräte in Anspruch nehmen können, falls Sie vorhanden sind,

☑	Berücksichtigung des Dolev-Yao-Angreifermodells
☑	Vermeidung von Single Points of Failure
☑	Sicheres, kontinuierliches Deployment
☑	Sichere externe Kommunikation berücksichtigt
☑	Flexible Unterstützung unterschiedlicher Autorisierungskonzepte
☑	Organisation versch. administrativer Gruppen
☑	Verschiedene Sicherheitsstufen
☑	Skalierbar
☑	Einfache Benutzbarkeit ohne Einschränkung der Sicherheit
☑	Unterstützung sicherer Datenhaltung
☑	Berücksichtigung der Leistungsheterogenität
☑	Entlastung von Kleinstgeräten
☑	Berücksichtigung der Heterogenität der Bedienelemente
☑	Geräte im Fokus statt Nutzer
☑	Schutz des übrigen Netzwerks bei Verlust eines Geräts
☑	Koordiniertes Abmelden eines Geräts
☑	Einrichtung / Wartung durch Experten und Nutzer berücksichtigt
☑	Kleine Konfigurationsänderung durch Nutzer
☑	Delegation von Rechten
☑	Weit verbreitete / akzeptierte Basistechnologie
☑	Sichere Protokollinteroperabilität
☑	Sichere Herstellerinteroperabilität
☑	Interoperable Ende-zu-Ende-Sicherheit

Tabelle 6.7: Zusammenfassung der Erfüllung der Anforderungen

andernfalls jedoch auch mit entsprechendem Mehraufwand selbst sichere Verbindungen zu anderen Teilnehmern aufbauen können.

Die Möglichkeit, delegierte Authentifizierung und synchrone Autorisierung auf verschiedene, starke Teilnehmer zu verteilen, garantiert ein hohes Maß an Skalierbarkeit und ermöglicht eine Aufteilung verschiedener Geräte in organisatorische Gruppen. Letzteres kann durch das entwickelte Autorisierungssystem durch die Verwendung unterschiedlicher Nutzer-Credential unterstrichen werden. Durch die Kombination von synchronen und asynchronen Autorisierern

kann ebenfalls flexibel eingestellt werden, ob die Konfiguration zentralisiert durch Experten oder dynamisch durch die Nutzer selbst erfolgen soll. Eine Mischform ist ebenfalls möglich. So lassen sich geschützte Bereiche einrichten, deren Konfiguration den Experten vorbehalten bleibt, während ein Nutzer auch Einstellungen an nicht kritischen Teilen einer Infrastruktur vornehmen kann.

Eine weitere Möglichkeit zur Umsetzung verschiedener Sicherheitsstufen ist die Aushandlung bzw. das Erfordern bestimmter Algorithmen oder größerer Mindestschlüssellängen. Außerdem unterstützt das Autorisierungssystem Angaben, die einem bestimmten Anfragesteller nur beschränkte Rechte auf eine Ressource einräumen, wie z.B. die Änderung der Raumtemperatur innerhalb bestimmter Grenzen. In infrastruktureller Hinsicht wurden alle die Autorisierung betreffenden Konzepte der ursprünglichen Spezifikationen vollständig übernommen, um der Anforderung an die flexible Unterstützung unterschiedlicher Autorisierungskonzepte gerecht zu werden.

Eine weitere Besonderheit insbesondere von Kleinstgeräten in intelligenten Umgebungen ist die Heterogenität und häufig das spärliche Vorhandensein von Nutzerschnittstellen. Um dennoch eine einfache Bedienung der Sicherheitsmechanismen zu gewährleisten, bindet DPWSec Multimediageräte als integralen Bestandteil der Sicherheitslösung ein. Dazu gehört auch, dass keine klassischen, nutzerzentrischen Authentifizierungsmethoden wie Nutzername-Passwort-Kombinationen genutzt, sondern gezielt auf die Fähigkeiten von Kleinstgeräten eingegangen wird, da in intelligenten Umgebungen die einzelnen Geräte Gegenstand der Kommunikation sind und nicht der Nutzer selbst.

Durch die Nutzung von Web Services bzw. von DPWS und der Web-Service-Security-Spezifikationssuite wird wie gefordert eine offene, vollständig unbeschränkte, weit verbreitete und akzeptierte Basistechnologie eingesetzt. Dies legt den Grundstein für eine sichere Herstellerinteroperabilität inklusive der geforderten Ende-zu-Ende-Sicherheit. Die Authentifizierungsinfrastruktur basiert nicht, wie so oft in der Literatur vorgeschlagen, auf vorinstallierten Zertifikaten und den dazugehörigen PKI und Trust-Hierarchien, da sich diese in der Vergangenheit nicht herstellerübergreifend behaupten konnten. Stattdessen werden Vertrauensbeziehungen zur Laufzeit und nach dem Deployment dynamisch aufgebaut. Dadurch bedarf es keines existierenden PKI und eine sichere Herstellerinteroperabilität ist gewährleistet.

Die Nutzung offener, weit verbreiteter Technologien begünstigt zudem eine sichere Protokollinteroperabilität, die jedoch in diesem Kapitel nicht nachgewiesen werden kann. Die dazugehörigen Aspekte werden in Kapitel 7 diskutiert. Dennoch handelt es sich bei der vorgestellten Lösung um ein Konzept auf Anwendungsebene. Das gewährleistet eine sichere externe Kommunikation, da die Mechanismen den Internet-Layer und damit IP-Router transparent passieren und zur Weiterleitung nicht entschlüsselt werden müssen. Gleichzeitig wird eine sichere Datenhaltung unterstützt, da die Nutzlast in verschlüsselter Form in der Anwendung vorliegt und ohne zusätzlichen Aufwand ebenso verschlüsselt persistiert werden kann.

DPWSec erfüllt demnach alle Anforderungen an eine Sicherheitsinfrastruktur für intelligente Umgebungen auf Basis von DPWS, die in Kapitel 5 auf Basis passender Angreifermodelle, der Literatur und den Szenarien erarbeitet wurden. Noch können jedoch keine Aussagen darüber getroffen werden, ob eine sichere Interoperabilität zwischen verschiedenen Kommunikationstechnologien möglich ist. Dies wird im folgenden Kapitel 7 untersucht. Eine weiterführende Diskussion von DPWSec erfolgt im Gesamtkontext dieser Arbeit in Kapitel 8.

Kapitel 7

Anwendbarkeit auf verschiedene Basistechnologien

Die im Verlauf dieser Arbeit mehrfach angesprochene Heterogenität verschiedener Komponenten in intelligenten Umgebungen bezieht sich nicht nur auf die Leistungsfähigkeit der Geräte, denn mit ihr gehen oft auch verschiedene Übertragungstechnologien einher. Ressourcenreiche Teilnehmer, die über eine permanente Stromversorgung verfügen und deren Standort selten wechselt, sind vor allem per Ethernet oder Powerline angebunden. Zur Minimierung des Verdrahtungsaufwandes oder weil es sich um mobile und somit batteriebetriebene Teilnehmer handelt, wird auch häufig WLAN eingesetzt.

Bei den zahllosen, eingesetzten Kleinstsensoren und -aktoren ist eine drahtgebundene Kommunikation oft nicht möglich, und WLAN erhöht den Leistungsbedarf enorm. Hier kommen Low-Power-Funktechnologien wie z.B. Bluetooth, ZigBEE und 6LoWPAN zum Einsatz. Verschiedene Geräteklassen kommunizieren demnach über verschiedene Übertragungstechnologien, die sichere Zusammenarbeit muss dennoch gewährleistet bleiben. In diesem Kapitel werden zunächst Betrachtungen zu einer sicheren Protokollinteroperabilität angestellt. Daraufhin werden die Bedingungen formuliert, unter denen sich die Kernkonzepte von DPWSec auf andere Layer-7-Protokolle übertragen lassen. Das Kapitel schließt mit einer kurzen, konzeptionellen Untersuchung, ob und in wie weit DPWSec eine sichere Interoperabilität zwischen verschiedenen Übertragungstechnologien zulässt, um eine sichere Übersetzung zwischen verschiedenen Anwendungsprotokollen zu ermöglichen.

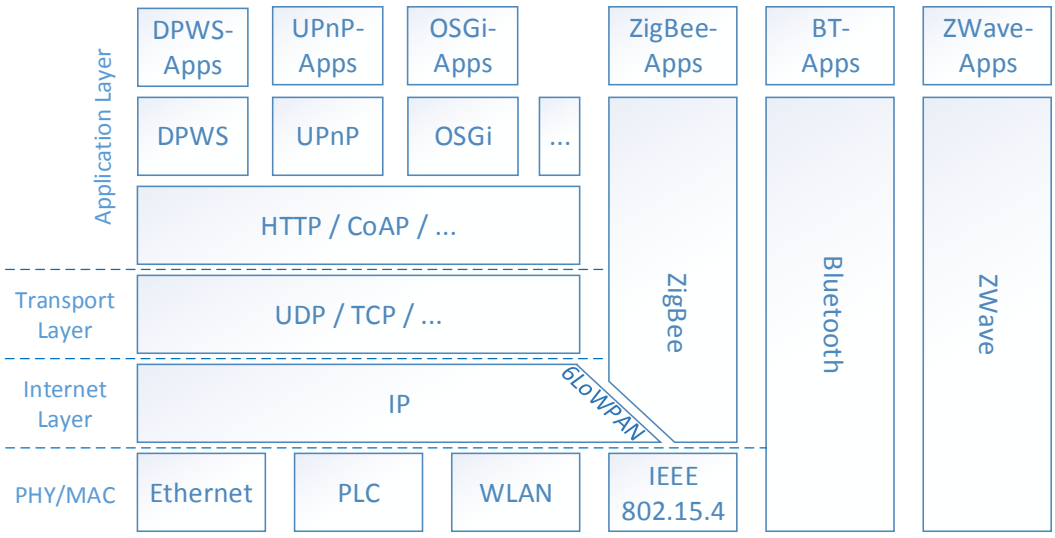


Abbildung 7.1: Möglicher Ausschnitt vorhandener Technologien in einer intelligenten Umgebung

7.1 Betrachtungen zur sicheren Protokollinteroperabilität

Wie erwähnt besteht eine Besonderheit intelligenter Umgebungen in der hohen Diversität der verwendeten Kommunikationstechnologien ([DZLT09]). Abbildung 7.1 zeigt einen möglichen Ausschnitt der eingesetzten Protokolle in einer intelligenten Umgebung. Auf der linken Seite ist zu erkennen, dass die Heterogenität insbesondere in den unteren beiden Ebenen des ISO/OSI-Modells besteht. So kommen häufig Ethernet oder PLC als drahtgebundene Technologien zum Einsatz. Als drahtlose Alternativen stehen WLAN und IEEE 802.15.4 bereit. Sie alle folgen dem ISO/OSI-Ebenen-Modell, wodurch – entsprechende Bridges vorausgesetzt – die technische Interoperabilität dieser Technologien schon durch den durchgängigen Einsatz des Internet Protokolls gewährleistet ist. Zender et al. bezeichnen dies in [ZDLT09] als „Heterogenität auf Netzwerkebene“. Sie unterscheiden diese von einer weiteren Dimension der Heterogenität, der „Heterogenität auf Dienstebene“. Geräte kapseln ihre Funktionalität in Dienste, um ihre Details zu verbergen. Kommen jedoch verschiedene dienstorientierte Kommunikationstechnologien zum Einsatz, so sind diese oft nicht zueinander kompatibel. Ein Client müsste demnach profunde Kenntnisse über verschiedene SOA-Technologien besitzen, um diese gezielt ansprechen zu kön-

nen. Das macht den eigentlichen Vorteil der Kapselung von Funktionalität in Dienste zunichte. Zender et al. postulieren in [ZDLT08] darüber hinaus, dass dieses Vorgehen für den Nutzer komplizierter wäre als eine klassische Gerätetreiberinstallation.

Für die Schaffung von Interoperabilität auf Dienstebene stehen drei Möglichkeiten zur Verfügung. Die einfachste Lösung besteht darin, eine Einigung auf eine Technologie auf Dienstebene anzustreben, ähnlich, wie sich das Internet Protokoll auf dem Internet-Layer durchgesetzt hat. Die verschiedenen SOAs haben jedoch teils so unterschiedliche Eigenschaften mit den jeweiligen Vor- und Nachteilen, dass eine Festlegung nicht zielführend scheint. Zudem existieren weitere relevante Technologien wie ZigBee, Bluetooth oder ZWave, die vollständige Protokollstapel implementieren, auf die Anwendungen unmittelbar aufsetzen. Die Dienstebene ist dabei nicht so leicht austauschbar wie in Protokollstapeln, die dem ISO/OSI-Layer-Modell entsprechen, auch wenn experimentelle Ansätze wie in [ABCR10] existieren, einen Web Service Layer z.B. oberhalb von Bluetooth einzurichten.

Da die Einigung auf eine gemeinsame Servicetechnologie nicht durchführbar scheint, muss demnach eine Strategie für die Übersetzung zwischen verschiedenen Technologien unter Beibehaltung der jeweiligen Ausführungssemantik eingesetzt werden. So ist es möglich, dass beispielsweise ein Bluetooth-Client eine Operation eines DPWS-Geräts aufruft. Zender et al. beschreiben in [ZDLT08], dass dafür häufig sogenannte *Bridge Nodes* eingesetzt werden. Es handelt sich dabei um unidirektionale Umsetzer von genau einer Technologie in genau eine andere. Dieses Konzept birgt den Nachteil, dass bei der Verwendung von t Technologien $\binom{t}{2}$ Bridge Nodes benötigt werden. Im Szenario in Abbildung 7.1 bedeutet dies bereits $\binom{6}{2} = 15$ verschiedene Bridge Nodes.

Diese Problemstellung führt zu der dritten Möglichkeit, Interoperabilität auf Dienstebene herzustellen. Sie liegt in der Einführung einer Metaebene, die Dienste abstrakt beschreibt. Somit bedarf es bei t vorhandenen Technologien lediglich t verschiedener Umsetzer und skaliert damit besser. Ein solches Konzept wurde im Rahmen des DFG Graduiertenkollegs 1424 „MuSAMA“ von Zender et al. in [ZLT10] vorgestellt. Sie stellen den *General Purpose Access Point* (GPAP) vor, der als zentrale Instanz in einem heterogenen Netzwerk sowohl die Netzwerk- als auch die Dienstheterogenität zwischen verschiede-

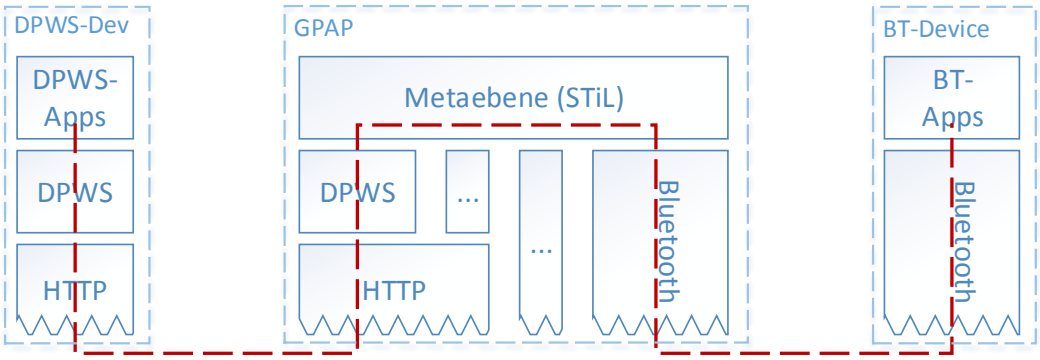


Abbildung 7.2: Herstellung der Dienstinteroperabilität über einen GPAP

nen Kommunikationstechnologien überwindet. Letzteres erreicht das Konzept durch Einführung der *"Service Technology-independant Language"* (STiL) zur abstrakten Beschreibung von Diensten. Hierdurch ergibt sich die Situation wie in Abbildung 7.2 angedeutet. Geräte auf Basis verschiedener Kommunikationstechnologien sprechen einander über einen GPAP an. Dieser übersetzt die Anfragen von der Quelltechnologie in die Metabeschreibung und von dort aus in die Zieltechnologie.

Die beiden Dimensionen der Heterogenität wirken sich auch auf die Möglichkeiten einer sicheren Interoperabilität aus. Das in Kapitel 6 beschriebene DPWSec ist vollständig auf Anwendungsebene implementiert und dadurch für die unteren Schichten des ISO/OSI-Ebenenmodells transparent. Die Heterogenität auf Netzwerkebene ist somit ohne weitere Aufwände überbrückt. Die Übersetzung des GPAP auf Dienstebene bietet die Möglichkeit, eine sichere Dienstinteroperabilität zwischen verschiedenen Kommunikationstechnologien zu gewährleisten. Dafür muss jedoch DPWSec auf andere Basistechnologien als DPWS übertragbar sein. Im Folgenden wird DPWSec daraufhin untersucht, welche Anforderungen es an eine Transport- bzw. Diensttechnologie stellt.

7.2 Voraussetzungen für die Übertragbarkeit

Das in Kapitel 6 vorgestellte Sicherheitsprofil DPWSec wurde mit dem Ziel entwickelt, einen Zwischenschritt hin zu einer vielseitigen Sicherheitsinfrastruktur für verteilte eingebettete Systeme darzustellen, wie sie in intelligenten Umgebungen anzutreffen sind. Somit handelt es sich bei DPWSec um eine Inkar-

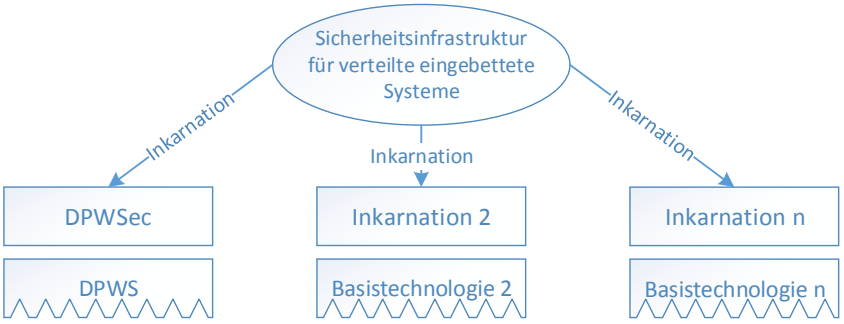


Abbildung 7.3: Verschiedene Inkarnationen der Sicherheitsinfrastruktur für verteilte eingebettete Systeme

nation der Sicherheitsinfrastruktur für die Basistechnologie DPWS. Damit die in Abschnitt 7.1 angestrebte sichere Dienstinteroperabilität realisiert werden kann, muss die entwickelte Infrastruktur durch weitere Inkarnationen auch auf andere Basistechnologien übertragbar sein, so, wie es in Abbildung 7.3 dargestellt ist. Im Folgenden wird untersucht, welche Eigenschaft eine Basistechnologie besitzen muss, um eine Inkarnation der vorgestellten Sicherheitsinfrastruktur zu unterstützen.

7.2.1 Discovery und Description

Für den Verbindungsaufbau mit anschließender Authentifizierung muss die Kommunikationsmiddleware den Teilnehmern zunächst Discovery- und Beschreibungsmechanismen bieten. Geräten muss es möglich sein, Typangaben zu publizieren. Idealerweise lassen sie sich über diese Typangaben bereits während eines Discovery-Vorgangs filtern. Clients müssen in der Lage sein, Geräte bestimmter Typen aufzufinden. Nur so lässt sich das dynamische Auffinden verschiedener Authentifizierer und UI-Authentifizierer umsetzen und synchrone Autorisierer finden auf diese Weise Geräte zum asynchronen Einholen von Nutzerentscheidungen.

Geräte sollten in der Lage sein, ihre die Sicherheit betreffenden Schnittstellen von den übrigen getrennt zu gruppieren. Diese organisatorische Gruppierung muss einen Mechanismus bieten, der die relevanten Sicherheitsparameter publizieren kann. Clients benötigen ein Mittel, um diese organisatorische Gruppierung zu identifizieren und die veröffentlichten Informationen zu beziehen. Hierzu muss es für den Client weiterhin möglich sein, aus der Menge der

angebotenen Operationen die jeweils richtige für einen Zweck zu bestimmen. Man bezeichnet dies auch als lokales Ressourcen- bzw. Operations-Discovery. Durch diese Mechanismen wird es Clients ermöglicht, z.B. zuständige Broker und Autorisierer, Authentifizierungsmethoden oder unterstützte Verschlüsselungsalgorithmen der Zielgeräte zu ermitteln.

Ein weiterer wichtiger Punkt besteht darin, dass sich die Sicherheitsparameter und damit auch die Beschreibung zur Laufzeit ändern können, zum Beispiel wenn ein Gerät einen Authenticator als Broker akzeptiert und dies öffentlich macht. Geräte müssen demnach in der Lage sein, ihre Beschreibung zur Laufzeit zu ändern und Clients müssen dies berücksichtigen. Für die eigentliche Funktionalität der Sicherheitsinfrastruktur ist es unerheblich, ob ein dezentraler, dynamischer Discovery-Ansatz genutzt wird oder eine zentrale Registry. Letzteres wirkt sich jedoch nachteilig auf die Robustheit aus, da dadurch ein Single-Point-of-Failure eingeführt wird.

7.2.2 Aufrufen von Operationen

An Operationsaufrufe und den damit einhergehenden Datenaustausch werden ebenfalls spezielle Anforderungen gestellt. Zunächst muss ein Operationsaufruf nach dem Request-Response-Kommunikationsmuster erfolgen. Dies ist das zum Zeitpunkt der Erstellung dieser Arbeit vorherrschende Muster und liegt beinahe allen im Internet der Dinge verbreiteten Protokollen wie IP, TCP, HTTP, Web Services, u.a. zugrunde. Ein zeitweises Blockieren eines Requests bis zum Eintreffen der Response muss sowohl von der Technologie als auch von der Implementierung unterstützt werden. Reine Publish-Subscribe-Ansätze wie zum Beispiel im Protokoll MQTT ([OAS14]) wurden nicht betrachtet. Es sei hier erwähnt, dass DPWSec asynchrone Nachrichten wie z.B. bei WS Eventing unterstützt, es wurden in dieser Arbeit jedoch keine Untersuchungen zur Tauglichkeit von Kommunikationstechnologien durchgeführt, die ausschließlich asynchrone Kommunikationsmuster anbieten.

Bei den eigentlichen Aufrufen ist es notwendig, dass sich die Nutzlast vom Header mit den entsprechenden Zusatzdaten trennen lässt. Andernfalls ist keine sichere Ende-zu-Ende-Interoperabilität möglich, da die beispielsweise verschlüsselte Nutzlast unverändert zwischen verschiedenen Protokollen durchgereicht werden muss. DPWSec nutzt die Möglichkeit des Basisprotokolls SOAP,

die zu übertragenden Daten zu strukturieren. Das bedeutet, dass Elemente einer Nachricht sowohl Unterelemente als auch Attribute besitzen können muss. Es ist von Vorteil, wenn eine Zieltechnologie diese Eigenschaft ebenfalls besitzt, andernfalls bedarf es einer Strategie zur Serialisierung und Deserialisierung strukturierter Daten in der Anwendung, was seinerseits wiederum zu Interoperabilitätsproblem führen kann.

Da es sich bei Chiffren und somit auch bei Signaturen immer um Binärdaten handelt, ist es schließlich notwendig, dass die Basistechnologie ihren Transport unterstützt. Handelt es sich bei dieser Technologie um ein ASCII-Protokoll, wie z.B. HTTP und SOAP, ist diese Eigenschaft in der Regel nicht gegeben. Auch hier ist es der Interoperabilität zuträglich, wenn es einen standardisierten Weg gibt, dieses Problem zu umgehen, wie beispielsweise die Verwendung von Base64-codierten Binärströmen in SOAP.

7.3 Zusammenfassung und Diskussion

Die notwendigen Eigenschaften, die eine Kommunikationstechnologie erfüllen muss, um als Basis für eine Inkarnation der entwickelten Sicherheitsinfrastruktur für verteilte eingebettete Systeme eingesetzt werden zu können, sind in Tabelle 7.1 zusammengefasst. Sie werden im Folgenden in aller Kürze am Beispiel von DPWS diskutiert.

Das Discovery von Geräten ist DPWS inhärent und unterstützt die Angabe von Gerätetypen sowie die gezielte Suche danach. Ein Dienst eines Geräts fungiert als logische Gruppierung von Operationen und bietet gleichzeitig die Möglichkeit, über Policies in seiner Dienstbeschreibung Verbindungsparameter wie unterstützte Algorithmen oder Authentifizierungsmethoden noch vor Verbindungsaufbau zur Verfügung zu stellen. Ein Mechanismus für das lokale Discovery von Ressourcen bzw. Operationen stellen Web Services in Form der *Port Types* bereit. Über diese Port Types lassen sich Funktionalitäten, wie z.B. eine bestimmte Authentifizierungsmethode, und die dazugehörigen Operationen anzeigen und auffinden. Eine dynamische Änderung einer Beschreibung zur Laufzeit des Geräts wird von DPWS nicht explizit unterstützt, jedoch ist der Fall auch nicht ausgeschlossen. Tatsächlich schreibt die Basisspezifikation vor, dass falls sich die Beschreibung eines Geräts ändert, dies durch Hochzählen

Discovery von Geräten
Filtern von Geräten während des Discovery
Discovery von Ressourcen (Operationen)
Publikation von Gerätetypen
Gruppierung von Operationen
Beschreibung einer Gruppierung
Beschreibung dynamisch änderbar
Request-Response-Pattern
Unterstützung von Blockierungen
Trennung von Header und Nutzlast
Transport strukturierter Daten
Binärdatentransport

Tabelle 7.1: Zusammenfassung der notwendigen Eigenschaften einer Kommunikationstechnologie, um als Basis für die entwickelte Sicherheitsinfrastruktur zu dienen

der Versionsnummer angezeigt werden muss. Das Detektieren von Änderungen ist über diesen Mechanismus möglich.

Web Services folgen grundsätzlich dem Request-Response-Pattern, sofern nicht asynchrones Messaging explizit durch die Verwendung von WS Eventing eingesetzt wird. Blockierende Aufrufe sind nicht ausgeschlossen und werden z.B. vom eingesetzten DPWS-Framework JMEDS unterstützt. SOAP trennt klar die eigentliche Nutzlast im Body eines SOAP-Envelopes von Verbindungsparametern, die im Header des Envelopes enthalten sind. Da SOAP auf XML basiert, unterstützt es nativ die Repräsentation und den Transport strukturierter Daten. XML ist zwar ein textbasiertes Protokoll, besitzt jedoch einen nativen, elementaren Datentyp zum Transport base64-codierter Binärdaten.

Jede weitere Kommunikationstechnologie, die diese Anforderungen erfüllt, kann als Basis für eine Inkarnation der entwickelten Sicherheitsinfrastruktur für verteilte, eingebettete Systeme genutzt werden. Eine Protokollübersetzung wie z.B. durch den GPAP vorausgesetzt, sind gesicherte Nachrichten auch zwischen verschiedenen Protokollen möglich. Ergänzt durch die Discovery- und Beschreibungsfähigkeiten wird es ermöglicht, dass direkte Vertrauensbeziehun-

...
☑ Weit verbreitete / akzeptierte Basistechnologie
☑ Sichere Protokollinteroperabilität
☑ Sichere Herstellerinteroperabilität
☑ Interoperable Ende-zu-Ende-Sicherheit

Tabelle 7.2: Zusammenfassung der Erfüllung der verbliebenen Anforderungen

gen zwischen zwei Geräten mit unterschiedlichen Kommunikationstechnologien hergestellt werden können. Das beinhaltet die geforderte interoperable Ende-zu-Ende-Sicherheit, da das Gerätepaar exklusive, gemeinsame geheime Schlüssel aufbauen kann. Auch die in Kapitel 6.3.5 beschriebenen Mechanismen zur optionalen Zuhilfenahme ressourcenreicher Teilnehmer lassen sich anwenden. Mit der Ermöglichung der protokollübergreifenden, direkten Authentifizierung und des anschließenden, sicheren Austauschs von Nachrichten werden zudem die delegierte Authentifizierung sowie die in Kapitel 6.3.6 beschriebenen Mechanismen zur Beantragung und Auslieferung von Autorisierungsentscheidungen ermöglicht.

Tabelle 7.2 zeigt die bisher nicht vollständig erfüllten Interoperabilitätsanforderungen aus Kapitel 5. Erfüllt eine Basistechnologie die in diesem Kapitel definierten Eigenschaften, so lassen sich die Mechanismen von DPWSec als weitere Inkarnation übertragen. Die Folge ist eine sichere Protokollinteroperabilität nicht nur auf Netzwerk-, sondern auch auf Dienstebene. Da zwei Geräte auch über Protokollgrenzen hinweg gemeinsames geheimes Schlüsselmaterial aufbauen können, ist zudem eine interoperable Ende-zu-Ende-Sicherheit möglich. Hierzu darf jedoch keine Protokollübersetzung der Nutzlast notwendig sein, da andernfalls ein Vermittler wie der GPAP die Nutzlast zum Zwecke der Übersetzung entschlüsseln müsste. Die in Kapitel 6.1 gezeigte weite Verbreitung von SOAP als Nachrichtenprotokoll legt jedoch nahe, dass keine Übersetzung der Nutzlast notwendig ist.

Kapitel 8

Zusammenfassung und Ausblick

Die vorliegende Arbeit leistet einen Beitrag zur Erhöhung der Sicherheit für verteilte, eingebettete Systeme, indem sie eine übertragungstechnologieunabhängige Sicherheitsinfrastruktur zur Wahrung der Kommunikationssicherheit und damit zur Erhöhung der Informationssicherheit vorschlägt.

Durch die Behandlung der Grundlagen und des Stands der Technik bei der sicheren Vernetzung klassischer Rechnernetze in Kapitel 2 stellt sich heraus, dass eine Sicherheitsinfrastruktur für verteilte, eingebettete Systeme auf Anwendungsebene implementiert sein muss, um ein hinreichendes Maß an Protokollunabhängigkeit bieten zu können.

Aus diesem Grund widmet sich Kapitel 3 dem Vergleich der Performance von Verschlüsselung und Signierung von Daten auf Anwendungsebene mit TLS auf Netzwerkebene als Stand der Technik. Das für den Vergleich erstellte WS Compact Security erhält ebenso Einzug in die entwickelte Sicherheitsinfrastruktur wie die entwickelte Möglichkeit, OOB-Kanäle zu übersetzen, wie es in Kapitel 4 beschrieben wurde und die Anforderungsanalyse aus Kapitel 5.

Die eigentliche Sicherheitsinfrastruktur wird anhand ihrer Inkarnation DPWSec in Kapitel 6 beschrieben, nachdem gezeigt werden konnte, dass der Stand der Technik bisher keine umfassende Lösung bietet, die allen Anforderungen gerecht werden kann. Es wird beschrieben, anhand welcher Methodik DPWSec aus den ursprünglichen Spezifikationen herausgearbeitet wurde, und das Ergebnis wird anhand konkreter Authentifizierungs- und Autorisierungsszenarien beschrieben. Schließlich zeigen Laufzeitmessungen des Prototypen die Möglichkeit des praktischen Einsatzes.

Da es nicht Ziel dieser Arbeit war, ein Sicherheitskonzept für DPWS zu entwickeln, sondern eine Sicherheitsinfrastruktur für verteilte eingebettete Systeme in intelligenten Umgebungen, zeigt Kapitel 7, dass das DPWSec zugrundeliegende Konzept unter bestimmten Bedingungen auf weitere Kommunikationstechnologien übertragbar ist und somit die Grundlage für eine sichere Protokollinteroperabilität darstellen kann.

Der Performance-Vergleich von WS Compact Security mit TLS in Kapitel 3 hat gezeigt, dass die RTTs bei der in dieser Arbeit vorgestellten Lösung bis zu Faktor 3 höher liegen als beim Stand der Technik, was als akzeptabel angesehen werden kann, um damit die Unabhängigkeit von TCP als Transportprotokoll zu erlangen. Der Grund für den Geschwindigkeitsvorteil von TLS ist in der Kompression der Daten zu suchen, was bei WS Compact Security keine Vorteile brachte. Es ist jedoch zu untersuchen, wie eine intelligente Kompression der SOAP-Nachrichten durch z.B. EXI diesen Vorteil von TLS zunichte machen kann und wie EXI mit der Verschlüsselung der Nutzlast bzw. einer eingebetteten Signatur zusammenspielt. Außerdem ist die weitere Entwicklung von TLS abzuwarten. Die kommende Version 1.3 liegt zum Zeitpunkt der Fertigstellung dieser Arbeit als Draft vor ([Res15]) und es zeichnet sich ab, dass Datenkompression in dieser Fassung vollständig verboten wird, da sie Side-Channel-Angriffe erlaubt, die die Vertraulichkeit der übertragenen Daten gefährden ([A+15]). Bleibt es bei dieser Entscheidung, muss der Vergleich zwischen der hier vorgestellten Lösung und TLS neu ausgeführt und bewertet werden, da TLS mit der Kompression mindestens einen Teil seines Geschwindigkeitsvorteils verliert. Schließlich hat der Autor dieser Arbeit zwar Sorgfalt bei der Implementierung des Prototyps walten lassen, es ist jedoch als wahrscheinlich anzusehen, dass diese Implementierung hinsichtlich ihrer Ausführungszeiten Optimierungspotential bietet, was den Abstand zwischen WS Compact Security und TLS weiter verkleinern würde.

Das vorgestellte Protokoll zur Übersetzung von OOB-Kanälen zum Zweck der indirekten Authentifizierung wurde in Kapitel 4.6 ausführlich diskutiert. Künftige Arbeiten betreffen zunächst Verfeinerungen. So wurde zur Umsetzung des Flacker-Plug-ins für den WS4D Mobile Authenticator die Android-Java-API genutzt, womit zuverlässig lediglich ein Symbol pro Sekunde übertragen werden konnte. Ein direkter Zugriff mit der verfügbaren C-API verspricht

weit höhere Symbolraten. Auch Ansätze wie der Rolling-Shutter-Mechanismus versprechen derzeit Übertragungsraten von bis zu 1.1kBd, was eine für das menschliche Auge unsichtbare Kommunikation mittels sichtbarem Licht zulässt ([DAP+12]). Auch wurde in dieser Arbeit lediglich die technische Machbarkeit demonstriert. Ergänzende Nutzerstudien müssen untersuchen, welche OOB Übertragungsmechanismen tatsächlich zu einer nutzerfreundlichen User-Experience führen, und dabei beachten, dass eventuell bestimmte Kombinationen von Mechanismen ungünstige Begleiterscheinungen hinsichtlich Nutzbarkeit oder Sicherheit mit sich bringen könnten. Schließlich bietet die Nutzung von Multimediakomponenten beim Verbindungsaufbau die Chance, das Ziel einer Authentifizierung festzulegen. So könnte ein Schalter eine Authentifizierung ohne Angabe eines Ziel initiieren, damit ein vertrauenswürdiges Multimediagerät dieses (z.B. durch eine Nutzerentscheidung) übernimmt. Diese Fragestellung bedarf jedoch vorsichtiger und gründlicher Untersuchungen, da die Auswirkungen auf die Sicherheit zu diesem Zeitpunkt nicht abgeschätzt werden können.

DPWSec, wie es in Kapitel 6 vorgestellt wird, wurde gezielt anhand einer Liste von Anforderungen entwickelt, die sich vor allem aus einer umfangreichen Szenariensammlung ergeben hat. Es existieren weitere, mögliche Eigenschaften, die in DPWSec nicht berücksichtigt werden, da sie nicht gefordert wurden, wie z.B. eine vollständige Multihop-Security (vgl. Abb. 6.5), sichere Gruppenkommunikation oder Authentifizierung über eine PKI wie dem Web-of-Trust. Während DPWSec seinen Funktionsumfang klar definiert, so sind dennoch die Erweiterungsmechanismen von Web Services gültig. Ein weiteres Sicherheitsprofil auf Basis von DPWSec könnte diese Mechanismen für andere Anwendungsgebiete nachrüsten. Eine gegebenenfalls gewünschte Abwärtskompatibilität zu DPWSec liegt jedoch in der Verantwortung des Entwicklers des Aufbauprofils.

Die in dieser Arbeit beschriebenen Software-Komponenten sind als Open-Source-Software mit einer Setup-Dokumentation unter [Ung15] zu finden. Das gesamte Proof-of-Concept wurde in Java implementiert, was einen vergleichsweise schnellen Funktionsnachweis erlaubte. Eine Implementierung z.B. in der Programmiersprache C erfordert einen höheren Aufwand, erlaubt jedoch eine performantere Implementierung. Eine studentische Arbeit [Stud1], die vom

Autor dieser Arbeit betreut wurde, legt nahe, dass von einer Implementierung in C ein Geschwindigkeitsvorteil der Berechnungen von zwei Größenordnungen (Faktor >100) gegenüber Java (beides ausgeführt auf einem Raspberry Pi) zu erwarten ist. Auf Basis einer solchen Implementierung für Tiny-Geräte ließe sich eine untere Grenze für Kleinstgeräte für DPWSec ableiten.

Schließlich war es Ziel dieser Arbeit, eine Sicherheitsinfrastruktur für verteilte, eingebettete Systeme zu entwickeln. Wie in Kapitel 6.3.6 erwähnt, ist Autorisierung jedoch kein infrastrukturelles Problem. Vielmehr muss ein Autorisierungssystem auf Basis zur Verfügung stehender Informationen eine Entscheidung treffen. Roman et al. erkennen in [RZL13], dass bisher kaum Neuerungen in der Zugriffsverwaltung für das Internet der Dinge existieren. Auf diesem Feld bedarf es demnach weiterer, sorgfältiger Arbeiten, um nutzerfreundliche, geräteorientierte Autorisierungskonzepte für intelligente Umgebungen zu erschaffen.

Kapitel 7 beschreibt die funktionalen Anforderungen an eine Basistechnologie, um DPWSec auf diese übertragen zu können. Eine konkrete, alternative Implementierung und Integration in ein Übersetzungskonzept wie den GPAP ([ZLT10]), erlauben detaillierte Untersuchungen zu praktischen Herausforderungen der sicheren Protokollinteroperabilität. Es wird ebenfalls erwähnt, dass DPWSec für das klassische Request-Response-Kommunikationsmuster entworfen wurde. Ansätze für allerkleinste Sensoren wie MQTT ([OAS14]) bieten Alternativen wie z.B. themenbezogenes Brokered-Publish-Subscribe. Es bedarf weiterer, gründlicher Untersuchungen, ob sich DPWSec auf alternative Kommunikationsparadigmen übertragen lässt.

Anhang A

Beispielnachrichten und Messwerte

An dieser Stelle werden Beispielnachrichten und Messwerte dargestellt, die im Verlauf der Arbeit referenziert werden. Um den Lesefluss zu begleiten, erscheinen sie hier in der Reihenfolge ihrer Erwähnung im Haupttext.

A.1 Zu Kapitel 3: Performance von Sicherheit auf Netzwerk- und Anwendungsebene

A.1.1 Beispiele für XML-Signature, XML-Encryption und ihre kompakten Repräsentationen

Das nachfolgende Listing zeigt eine sehr einfache SOAP-Nachricht, inklusive dem HTTP-Header, der ihr beim Versand durch das SOAP-over-HTTP-Binding vorausgeht. Der HTTP-Header ist rot markiert, die eigentliche Nutzlast blau. Zeilenumbrüche und Einrückungen wurden zur besseren Lesbarkeit hinzugefügt.

```
1 HTTP/1.1 200 OK Server: gSOAP/2.8 Content-Type: text/xml; charset=utf-8
      Content-Length: 236 Connection: keep-alive\n\n
2
3 <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
4   <SOAP-ENV:Body>
5     <ns:single-string-echoResponse xmlns:ns="ws4d:ptest">
6       <ns:out>01234</ns:out>
7     </ns:single-string-echoResponse>
8   </SOAP-ENV:Body>
9 </SOAP-ENV:Envelope>
```

Listing A.1: Beispielnachricht für den Vergleich von Verschlüsselung auf Transport- und auf Anwendungsebene

Das nachfolgende Listing zeigt die entstehende Nachricht, wenn die Nachricht aus Listing A.1 mittels WS Security signiert und die Nutzlast verschlüsselt wird. Der Overhead zur Einbettung der Signatur ist blau und rot markiert, letzteres zeigt das `<ds:Reference>`-Element, das für jedes zu signierende Element der Nachricht angelegt wird. Grün markiert ist der Overhead der für die Einbettung der verschlüsselten Payload (violett) eingeführt wird. Zeilenumbrüche und Einrückungen wurden zur besseren Lesbarkeit hinzugefügt.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis
        -200401-wss-wssecurity-secext-1.0.xsd" SOAP-ENV:mustUnderstand="1">
      <xenc:ReferenceList xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
        <xenc:DataReference URI="Body"></xenc:DataReference>
      </xenc:ReferenceList>
      <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
        <ds:SignedInfo>
          <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-
              exc-c14n#"></ds:CanonicalizationMethod>
          <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#
              hmac-sha1"></ds:SignatureMethod>
          <ds:Reference URI="#Body">
            <ds:Transforms>
              <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#
                  "></ds:Transform>
            </ds:Transforms>
            <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1
                "></ds:DigestMethod>
            <ds:DigestValue>YA/1MWdGnWy+LxH2sdtowiBak2o=</ds:DigestValue>
          </ds:Reference>
        </ds:SignedInfo>
        <ds:SignatureValue>DsC7FivwVeVYJBgkwMVwNZjIQLc=</ds:SignatureValue>
      </ds:Signature>
    </wsse:Security>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
      wssecurity-utility-1.0.xsd" wsu:Id="Body">
    <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" Id="
        Body">
      <ds:KeyInfo>
        <wsse:SecurityTokenReference>
          <wsse:KeyIdentifier valueType="http://www.ws4d.org/ws4d-sec/2011/08/
              rc4-symmetric-key" EncodingType="#Base64Binary">
```

```
28              ZHVtbXk=
29            </wsse:KeyIdentifier>
30          </wsse:SecurityTokenReference>
31        </ds:KeyInfo>
32        <xenc:CipherData>
33          <xenc:CipherValue>MHXBD8eU39bFimFQqP6di1KedCheQQWpl0WxKLjQGN2jHy3/
                E0bMl1c8Usx3ymOg2RToEGw0IkA75uMQeGzEse7fRozOXF7UU1Pd1W04Ew+
                iduSori5/czCxIfPEX0zV0BzV+xLN2cLrFn0qBZh3Y8NWR6CfvPbc</
                xenc:CipherValue>
34        </xenc:CipherData>
35      </xenc:EncryptedData>
36    </SOAP-ENV:Body>
37  </SOAP-ENV:Envelope>
```

Listing A.2: Beispielnachricht für Verwendung von WS-Security (1862 Byte, ohne HTTP-Header)

Das nachfolgende Listing zeigt ein Beispiel für eine XML-Signature einer Nachricht, wobei die vier Elemente `<RelatesTo>`, `<Action>`, `<Timestamp>` und `<Body>` in die Signaturbildung eingehen. Zeilenumbrüche und Einrückungen wurden zur besseren Lesbarkeit hinzugefügt.

```
1  <ds:Signature xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
2    <ds:SignedInfo>
3      <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-
           c14n#" />
4      <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"
            />
5      <ds:Reference URI="#relatesto">
6        <ds:Transforms>
7          <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
8            <ec:InclusiveNamespaces PrefixList="i" xmlns:ec="http://www.w3.org
                 /2001/10/xml-exc-c14n#"/>
9          </ds:Transform>
10       </ds:Transforms>
11       <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
12       <ds:DigestValue>ODE3NDkyNzI5</ds:DigestValue>
13     </ds:Reference>
14   </ds:SignedInfo>
15   <ds:SignedInfo>
16     <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-
           c14n#" />
17     <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"
            />
18     <ds:Reference URI="#action">
19       <ds:Transforms>
20         <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
21           <ec:InclusiveNamespaces PrefixList="i" xmlns:ec="http://www.w3.org
                 /2001/10/xml-exc-c14n#"/>
22         </ds:Transform>
```

```
23      </ds:Transforms>
24      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
25      <ds:DigestValue>1D234D5y6z7F</ds:DigestValue>
26    </ds:Reference>
27  </ds:SignedInfo>
28  <ds:SignedInfo>
29    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-
          c14n#" />
30    <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"
          />
31    <ds:Reference URI="#timestamp">
32      <ds:Transforms>
33        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
34          <ec:InclusiveNamespaces PrefixList="i" xmlns:ec="http://www.w3.org
                /2001/10/xml-exc-c14n#"/>
35        </ds:Transform>
36      </ds:Transforms>
37      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
38      <ds:DigestValue>1a2B4c5yDzeF</ds:DigestValue>
39    </ds:Reference>
40  </ds:SignedInfo>
41  <ds:SignedInfo>
42    <ds:CanonicalizationMethod Algorithm="http://www.w3.org/2001/10/xml-exc-
          c14n#" />
43    <ds:SignatureMethod Algorithm="http://www.w3.org/2000/09/xmldsig#rsa-sha1"
          />
44    <ds:Reference URI="#body">
45      <ds:Transforms>
46        <ds:Transform Algorithm="http://www.w3.org/2001/10/xml-exc-c14n#">
47          <ec:InclusiveNamespaces PrefixList="i" xmlns:ec="http://www.w3.org
                /2001/10/xml-exc-c14n#"/>
48        </ds:Transform>
49      </ds:Transforms>
50      <ds:DigestMethod Algorithm="http://www.w3.org/2000/09/xmldsig#sha1" />
51      <ds:DigestValue>aDb3cDdyezf6</ds:DigestValue>
52    </ds:Reference>
53  </ds:SignedInfo>
54  <ds:SignatureValue>ru5Ef76xGz5Y5IB2iAzDuMvR5Tg=</ds:SignatureValue>
55  <ds:KeyInfo>
56    <wsse:SecurityTokenReference>
57      <wsse:KeyIdentifier>Dx42/9g=</wsse:KeyIdentifier>
58    </wsse:SecurityTokenReference>
59  </ds:KeyInfo>
60</ds:Signature>
```

Listing A.3: Beispiel für eine XML-Signature

Das nachfolgende Listing zeigt die Repräsentation der XML-Signature aus dem Beispiel in Listing A.3 als WSDD Compact Signature. Zeilenumbrüche und Einrückungen wurden zur besseren Lesbarkeit hinzugefügt.

```
<d:Sig   xmlns:d="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01"
         Scheme="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01/rsa"
         KeyId="Dx42/9g="
         Refs="relatesto,action,timestamp,body"
         PrefixList="i"
         Sig="ru5Ef76xGz5Y5IB2iAzDuMvR5Tg=" />
```

Listing A.4: Beispiel für eine WSDD Compact Signature

Das nachfolgende Listing zeigt ein Beispiel zur Verwendung von XML-Encryption zur Einbettung von Chiffren in eine SOAP-Nachricht. Zeilenumbrüche und Einrückungen wurden zur besseren Lesbarkeit hinzugefügt.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis
        -200401-wss-wssecurity-secext-1.0.xsd" SOAP-ENV:mustUnderstand="1">
      <xenc:ReferenceList xmlns:xenc="http://www.w3.org/2001/04/xmlenc#">
        <xenc:DataReference URI="Body"></xenc:DataReference>
      </xenc:ReferenceList>
    </wsse:Security>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
      xmlns:wsu="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
      wssecurity-utility-1.0.xsd" wsu:Id="Body">
    <xenc:EncryptedData xmlns:xenc="http://www.w3.org/2001/04/xmlenc#" Id="
        Body">
      <ds:KeyInfo>
        <wsse:SecurityTokenReference>
          <wsse:KeyIdentifier valueType="http://www.ws4d.org/ws4d-sec/2011/08/
              rc4-symmetric-key" EncodingType="#Base64Binary">
            ZHVtbXk=
          </wsse:KeyIdentifier>
        </wsse:SecurityTokenReference>
      </ds:KeyInfo>
      <xenc:CipherData>
        <xenc:CipherValue>
            pJ8uVIshC7UglXLfoChGRYu5pt9PrcrVNjQT58UAMGyNGDZn7kxtaRfllROciJnVOC
            K73/IFDEJR12Mvz7hCZes+its7Y+GuH7oUnC9tUkiqr1DjdfAOCiIiiBotXzzWpjh
            YfmXkqDo=</xenc:CipherValue>
      </xenc:CipherData>
    </xenc:EncryptedData>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listing A.5: Beispiel für die Verwendung von XML-Encryption

Das nachfolgende Listing zeigt die kompakte Repräsentation der Einbettung von Chiffren in die SOAP-Nachricht aus Listing A.5. Zeilenumbrüche und Einrückungen wurden zur besseren Lesbarkeit hinzugefügt.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <wsse:Security xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis
        -200401-wss-wssecurity-secext-1.0.xsd">
      <d:Enc  KeyId="ZHVtbXk="
              Refs="Body"
              Scheme="http://www.ws4d.org/ws4d-sec/2011/08/rc4" />
    </wsse:Security>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body wsu:ID="body">
    <d:EncryptedData xmlns:d="http://docs.oasis-open.org/ws-dd/ns/discovery
        /2009/01" Id="bodyID">
      <d:CipherData>
          pJ8uVIshC7UglXLfoChGRYu5pt9PrcrVNjQT58UAMGyNGDZn7kxtaRfllROciJnVOC
          K73/IFDEJR12Mvz7hCZes+its7Y+GuH7oUnC9tUkiqr1DjdfAOCiIiiBotXzzWpjh
          YfmXkqDo=</d:CipherData>
    </d:EncryptedData>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listing A.6: Kompakte Repräsentation der Einbettung von Chiffren

A.1.2 Test-Nachrichten für Performance-Messungen

Die Listings A.7 und A.8 zeigen die Request und Response für den Testfall 1 x 5 Zeichen. Zeilenumbrüche und Einrückungen wurden zur besseren Lesbarkeit hinzugefügt.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <ns:single-string-echo xmlns:ns="ws4d:ptest">
      <ns:in>01234</ns:in>
    </ns:single-string-echo>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listing A.7: Test-Nachricht an Echo Service, 1 String à 5 Zeichen (218 Byte, 134 Byte Nutzlast)

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <ns:single-string-echoResponse xmlns:ns="ws4d:ptest">
      <ns:out>01234</ns:out>
    </ns:single-string-echoResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listing A.8: Antwort auf Test-Nachricht an Echo Service, 1 String à 5 Zeichen (236 Byte, 152 Byte Nutzlast)

Die Listings A.9 und A.10 zeigen die Request und Response für den Testfall 1 x 50 Zeichen. Zeilenumbrüche und Einrückungen wurden zur besseren Lesbarkeit hinzugefügt.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <ns:single-string-echo xmlns:ns="ws4d:ptest">
      <ns:in>01234567890123456789012345678901234567890123456789</ns:in>
    </ns:single-string-echo>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listing A.9: Test-Nachricht an Echo Service, 1 String à 50 Zeichen (263 Byte), 179 Byte Nutzlast

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <ns:single-string-echoResponse xmlns:ns="ws4d:ptest">
      <ns:out>01234567890123456789012345678901234567890123456789</ns:out>
    </ns:single-string-echoResponse>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listing A.10: Antwort auf Test-Nachricht an Echo Service, 1 String à 50 Zeichen (281 Byte, 197 Byte Nutzlast)

Die Listings A.11 und A.12 zeigen die Request und Response für den Testfall 10 x 5 Zeichen. Zeilenumbrüche und Einrückungen wurden zur besseren Lesbarkeit hinzugefügt.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <ns:multiple-string-echo xmlns:ns="ws4d:ptest">
      <ns:in>
        <ns:p01>01234</ns:p01>
        <ns:p02>01234</ns:p02>
        <ns:p03>01234</ns:p03>
        <ns:p04>01234</ns:p04>
        <ns:p05>01234</ns:p05>
        <ns:p06>01234</ns:p06>
        <ns:p07>01234</ns:p07>
        <ns:p08>01234</ns:p08>
        <ns:p09>01234</ns:p09>
```

```
14          <ns:p10>01234</ns:p10>
15        </ns:in>
16      </ns:multiple-string-echo>
17    </SOAP-ENV:Body>
18 </SOAP-ENV:Envelope>
```

Listing A.11: Test-Nachricht an Echo Service, 10 Strings à 5 Zeichen (437 Byte, 353 Byte Nutzlast)

```
1 <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
2   <SOAP-ENV:Body>
3     <ns:multiple-string-echoResponse xmlns:ns="ws4d:ptest">
4       <ns:out>
5         <ns:p01>01234</ns:p01>
6         <ns:p02>01234</ns:p02>
7         <ns:p03>01234</ns:p03>
8         <ns:p04>01234</ns:p04>
9         <ns:p05>01234</ns:p05>
10        <ns:p06>01234</ns:p06>
11        <ns:p07>01234</ns:p07>
12        <ns:p08>01234</ns:p08>
13        <ns:p09>01234</ns:p09>
14        <ns:p10>01234</ns:p10>
15      </ns:out>
16    </ns:multiple-string-echoResponse>
17  </SOAP-ENV:Body>
18 </SOAP-ENV:Envelope>
```

Listing A.12: Antwort auf Test-Nachricht an Echo Service, 10 Strings à 5 Zeichen (455 Byte, 371 Byte Nutzlast)

Kompressionsstufe	Testfall	Chunk - Größe				Nachrichtengröße	Kompressionsrate
		1024	2048	4096	8192		
0	1	30,27	30,34	30,19	30,22	811	—
	2	31,08	31,09	31,03	31,09	871	
	3	41,84	41,27	41,15	41,29	1103	
1	1	36,96	37,10	37,02	37,04	550	32,18%
	2	38,01	38,06	37,94	38,07	599	31,23%
	3	49,71	49,15	48,85	48,91	785	28,83%
2	1	36,94	37,33	36,99	36,93	549	32,31%
	2	37,95	38,21	37,86	37,96	597	31,46%
	3	49,90	49,08	48,86	48,93	783	29,01%
3	1	36,98	37,28	37,03	36,88	549	32,31%
	2	38,13	38,18	37,95	37,97	597	31,46%
	3	50,20	49,33	49,01	49,01	783	29,01%
4	1	37,63	37,72	37,51	37,51	541	33,29%
	2	38,52	38,78	38,54	38,54	589	32,38%
	3	50,47	49,84	49,89	49,59	775	29,74%
5	1	37,62	37,98	37,63	37,64	542	33,17%
	2	38,77	38,87	38,53	38,67	591	32,15%
	3	50,56	49,87	49,74	50,07	777	29,56%
6	1	37,52	37,82	37,62	37,62	542	33,17%
	2	38,70	38,95	38,54	38,66	591	32,15%
	3	50,61	49,82	49,89	49,75	777	29,56%
7	1	37,69	37,79	37,63	37,50	542	33,17%
	2	38,71	38,81	38,63	38,75	591	32,15%
	3	50,87	50,24	49,77	49,75	777	29,56%
8	1	37,69	37,91	37,57	37,58	542	33,17%
	2	38,72	38,93	38,60	38,64	591	32,15%
	3	50,52	49,92	49,88	50,10	777	29,56%
9	1	37,71	37,96	37,47	37,57	542	33,17%
	2	38,68	38,93	38,72	38,74	591	32,15%
	3	50,61	50,00	49,96	49,94	777	29,56%

Tabelle A.1: Round Trip Times und Nachrichtengrößen bei Verwendung von Deflate-Kompression bei Nutzung der RC4-Stromchiffre

Kompressionsstufe	Testfall	Chunk - Größe				Nachrichtengröße	Kompressionsrate
		1024	2048	4096	8192		
0	1	39,07	38,85	38,77	38,93	833	
	2	40,07	40,12	40,34	40,36	897	
	3	49,69	49,25	49,20	49,26	1129	
1	1	45,43	45,41	45,25	45,44	572	31,33%
	2	47,09	47,09	47,23	47,06	619	30,99%
	3	58,00	56,91	56,87	56,95	807	28,52%
2	1	45,36	45,58	45,35	45,35	569	31,69%
	2	46,95	47,21	47,24	47,10	617	31,22%
	3	57,70	57,40	57,09	57,00	805	28,70%
3	1	45,44	45,68	45,48	45,35	571	31,45%
	2	46,91	47,13	47,12	47,23	618	31,10%
	3	58,02	57,35	56,97	56,94	806	28,61%
4	1	46,01	46,04	45,95	46,03	562	32,53%
	2	47,80	47,86	48,05	48,10	611	31,88%
	3	58,40	57,77	57,51	57,50	797	29,41%
5	1	45,99	46,32	45,97	45,97	563	32,41%
	2	47,97	48,06	48,15	48,07	612	31,77%
	3	58,63	58,21	57,67	57,66	800	29,14%
6	1	46,12	46,16	46,01	46,05	563	32,41%
	2	47,96	48,22	48,23	48,17	613	31,66%
	3	58,95	57,97	57,67	57,98	800	29,14%
7	1	46,01	46,29	46,03	45,95	563	32,41%
	2	48,06	47,99	48,14	48,26	612	31,77%
	3	58,63	58,07	57,64	57,70	799	29,23%
8	1	46,16	46,37	45,91	46,16	563	32,41%
	2	47,98	48,14	48,02	48,36	612	31,77%
	3	58,97	57,78	57,55	57,77	800	29,14%
9	1	46,19	46,31	46,17	45,96	562	32,53%
	2	48,01	48,02	48,24	48,22	611	31,88%
	3	58,62	57,83	57,75	57,90	798	29,32%

Tabelle A.2: Round Trip Times und Nachrichtengrößen bei Verwendung von Deflate-Kompression bei Nutzung der AES-CBC-Blockchiffre

A.2 Zu Kapitel 6.4: Prototypische Implementierung

Im Folgenden finden sich die Nachrichten, die zum Zweck der Operationsaufrufe versendet wurden, auf die sich die Messwerte aus Kapitel 6.4 beziehen.

```
<s12:Envelope xmlns:dpws="http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01"
    xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://www.w3.org/2005/08/addressing"
    xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
        wssecurity-secext-1.0.xsd">
  <s12:Header>
    <wsa:Action
        n1:Id="HID1"
        xmlns:n1="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01">
      http://www.ws4d.org/LightBulbService/LightBulbState
    </wsa:Action>
    <wsa:MessageID
        n2:Id="HID2"
        xmlns:n2="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01">
      urn:uuid:86a97340-a3ce-11e4-bf41-efd7a498dbe3
    </wsa:MessageID>
    <wsa:To n3:Id="HID3"
        xmlns:n3="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01">
      http://192.168.2.103:58826/org/LightBulbService
    </wsa:To>
  </s12:Header>
  <s12:Body />
</s12:Envelope>
```

Listing A.13: Ungesicherte Nachricht - Anfrage

```
<s12:Envelope xmlns:dpws="http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01"
    xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://www.w3.org/2005/08/addressing"
    xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
        wssecurity-secext-1.0.xsd">
  <s12:Header>
    <wsa:Action
        n1:Id="HID1"
        xmlns:n1="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01">
      http://www.ws4d.org/LightBulbService/LightBulbStateResponse
    </wsa:Action>
    <wsa:RelatesTo
        n2:Id="HID2"
        xmlns:n2="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01">
      urn:uuid:86a97340-a3ce-11e4-bf41-efd7a498dbe3
    </wsa:RelatesTo>
    <wsa:To
        n3:Id="HID3"
```

```
18          xmlns:n3="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01">
19        http://www.w3.org/2005/08/addressing/anonymous
20      </wsa:To>
21    </s12:Header>
22    <s12:Body>
23      <n4:state
24          n6:type="n5:string"
25          xmlns:n4="http://www.ws4d.org"
26          xmlns:n5="http://www.w3.org/2001/XMLSchema" xmlns:n6="http://www.w3.
                org/2001/XMLSchema-instance">
27        off
28      </n4:state>
29    </s12:Body>
30 </s12:Envelope>
```

Listing A.14: Ungesicherte Nachricht - Antwort

```
1 <s12:Envelope
2     xmlns:dpws="http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01"
3     xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
4     xmlns:wsa="http://www.w3.org/2005/08/addressing"
5     xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
          wssecurity-secext-1.0.xsd">
6   <s12:Header>
7     <wsa:Action
8         n1:Id="HID1"
9         xmlns:n1="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01">
10      http://www.ws4d.org/LightBulbService/LightBulbSwitch
11    </wsa:Action>
12    <wsa:MessageID
13        n2:Id="HID2"
14        xmlns:n2="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01">
15      urn:uuid:c22c5d90-a3d0-11e4-80d0-f35e35c93daa
16    </wsa:MessageID>
17    <wsa:To
18        n3:Id="HID3"
19        xmlns:n3="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01">
20      http://192.168.2.103:34903/org/LightBulbService
21    </wsa:To>
22    <wsa:Security>
23      <wsa:Enc
24          KeyId="123345678"
25          Refs=""
26          Scheme="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01/aes">
27      </wsa:Enc>
28      <wsa:Sig
29          Scheme="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01/rsa"
30          Refs="HID1␣HID2␣HID3"
31          Sig="YH1BKgE3rzFtnJErSBUWZIeB6i7P58eVWEFk9Q3JCDI=">
32      </wsa:Sig>
33    </wsa:Security>
34  </s12:Header>
```

```
35   <s12:Body>
36     <n4:state
37         n6:type="n5:string"
38         xmlns:n4="http://www.ws4d.org"
39         xmlns:n5="http://www.w3.org/2001/XMLSchema"
40         xmlns:n6="http://www.w3.org/2001/XMLSchema-instance">
41       on
42     </n4:state>
43   </s12:Body>
44 </s12:Envelope>
```

Listing A.15: Signierte Nachricht - Anfrage

```
1 <s12:Envelope
2     xmlns:dpws="http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01"
3     xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
4     xmlns:wsa="http://www.w3.org/2005/08/addressing"
5     xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
          wssecurity-secext-1.0.xsd">
6   <s12:Header>
7     <wsa:Action
8         n1:Id="HID1"
9         xmlns:n1="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01">
10      http://www.ws4d.org/LightBulbService/LightBulbSwitchResponse
11    </wsa:Action>
12    <wsa:RelatesTo
13        n2:Id="HID2"
14        xmlns:n2="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01">
15      urn:uuid:87aa1a60-a3ce-11e4-bf44-efd7a498dbe3
16    </wsa:RelatesTo>
17    <wsa:To
18        n3:Id="HID3"
19        xmlns:n3="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01">
20      http://www.w3.org/2005/08/addressing/anonymous
21    </wsa:To>
22    <wsa:Security>
23      <wsa:Enc
24          KeyId="123345678"
25          Refs=""
26          Scheme="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01/aes">
27      </wsa:Enc>
28      <wsa:Sig
29          Scheme="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01/rsa"
30          Refs="HID1␣HID2␣HID3"
31          Sig="osgGcuf0EPB00xEYtCwh6TfoY+UwwJc1nDflwkpchkM=">
32      </wsa:Sig>
33    </wsa:Security>
34  </s12:Header>
35  <s12:Body>
36    <n4:state
37        n6:type="n5:string"
38        xmlns:n4="http://www.ws4d.org"
```

```
39          xmlns:n5="http://www.w3.org/2001/XMLSchema"
40          xmlns:n6="http://www.w3.org/2001/XMLSchema-instance">
41        on
42      </n4:state>
43    </s12:Body>
44 </s12:Envelope>
```

Listing A.16: Signierte Nachricht - Antwort

```
1 <s12:Envelope
2     xmlns:dpws="http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01"
3     xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
4     xmlns:wsa="http://www.w3.org/2005/08/addressing"
5     xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
          wssecurity-secext-1.0.xsd">
6   <s12:Header>
7     <wsa:Action
8         n1:Id="HID1"
9         xmlns:n1="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01">
10      http://www.ws4d.org/LightBulbService/LightBulbSwitch
11    </wsa:Action>
12    <wsa:MessageID
13        n2:Id="HID2"
14        xmlns:n2="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01">
15      urn:uuid:86e60710-a3ce-11e4-bf42-efd7a498dbe3
16    </wsa:MessageID>
17    <wsa:To
18        n3:Id="HID3"
19        xmlns:n3="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01">
20      http://192.168.2.103:58826/org/LightBulbService
21    </wsa:To>
22    <wsa:Security>
23      <wsa:Enc
24          KeyId="123345678"
25          Refs=""
26          Scheme="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01/aes">
27      </wsa:Enc>
28      <wsa:Sig
29          Scheme="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01/rsa"
30          Refs="HID1␣HID2␣HID3"
31          Sig="sPnbbRjCbzyZzDX1rtCrpvoQSkl0N/oCcYeYR9Yamhk=">
32      </wsa:Sig>
33    </wsa:Security>
34  </s12:Header>
35  <s12:Body>
36    OS5BVSOjAr1G4KQ7+s3Q74Iwk+ahqc2DY4mn05bEDX7K6hFPRf+
          za7hBhzpGUlUNSXbu6xeVqO4oPVaXn9GtBawoYa2FpISHaO3cof9wVs4prHcqwNNkZg8/
          L8EEg5TpOCqsUYoD8oaJC9THjpmbE5EPojAMchDCaGfpyDDO5ifU81Ayn1Thy1ucXWT+
          HdKR7bo8HACaUcxrI/M7SZ+EoOaeehgMhBO1kN6+zoc/GNE=
37  </s12:Body>
38 </s12:Envelope>
```

Listing A.17: Signierte und verschlüsselte Nachricht - Anfrage

```
<s12:Envelope
    xmlns:dpws="http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01"
    xmlns:s12="http://www.w3.org/2003/05/soap-envelope"
    xmlns:wsa="http://www.w3.org/2005/08/addressing"
    xmlns:wsse="http://docs.oasis-open.org/wss/2004/01/oasis-200401-wss-
        wssecurity-secext-1.0.xsd">
  <s12:Header>
    <wsa:Action
        n1:Id="HID1"
        xmlns:n1="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01">
      http://www.ws4d.org/LightBulbService/LightBulbSwitchResponse
    </wsa:Action>
    <wsa:RelatesTo
        n2:Id="HID2"
        xmlns:n2="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01">
      urn:uuid:86e60710-a3ce-11e4-bf42-efd7a498dbe3
    </wsa:RelatesTo>
    <wsa:To
        n3:Id="HID3"
        xmlns:n3="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01">
      http://www.w3.org/2005/08/addressing/anonymous
    </wsa:To>
    <wsa:Security>
      <wsa:Enc
          KeyId="123345678"
          Refs=""
          Scheme="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01/aes">
      </wsa:Enc>
      <wsa:Sig
          Scheme="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01/rsa"
          Refs="HID1␣HID2␣HID3"
          Sig="Nal9SQz+WSApSUIf9DqK32E9IYA15x5wcFWPws3ILk4=">
      </wsa:Sig>
    </wsa:Security>
  </s12:Header>
  <s12:Body>
    OS5BVSOjAr1G4KQ7+s3Q74Iwk+ahqc2DY4mn05bEDX7K6hFPRf+
        za7hBhzpGUlUNSXbu6xeVqO4oPVaXn9GtBawoYa2FpISHaO3cof9wVs4prHcqwNNkZg8/
        L8EEg5TpOCqsUYoD8oaJC9THjpmbE5EPojAMchDCaGfpyDDO5ifU81Ayn1Thy1ucXWT+
        HdKR7bo8HACaUcxrI/M7SZ+EoOaeehgMhBO1okN6+zoc/GNE=
  </s12:Body>
</s12:Envelope>
```

Listing A.18: Signierte und verschlüsselte Nachricht - Antwort

Anhang B

Profil

Dieser Abschnitt beschreibt technisch und detailliert das entstandene Profil DPWSec, das funktional in Kapitel 6 beschrieben wird. Es gelten die Konventionen aus RFC 2119 [Bra97] hinsichtlich der Begriffe MÜSSEN, DÜRFEN NICHT, SOLLTEN, SOLLTEN NICHT, DÜRFEN, EMPFOHLEN und OPTIONAL.

B.1 Namespaces

In DPWSec werden die folgenden Namespaces mit den angegebenen Prefixes verwendet.

Prefix	Namespace
auth	http://docs.oasis-open.org/wsfed/authorization/200706
fed	http://docs.oasis-open.org/wsfed/federation/200706
saml	urn:oasis:names:tc:SAML:2.0:assertion
ws4d	http://www.ws4d.org/
wsa	http://www.w3.org/2005/08/addressing
wsc	http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512
wsp	http://www.w3.org/ns/ws-policy
wst	http://docs.oasis-open.org/ws-sx/ws-trust/200512

Tabelle B.1: Zusammenfassung der erforderlichen Eigenschaften

B.2 Angabe des Profils

Die Unterstützung von DPWSec wird angezeigt, indem die in Abschnitt B.3 angegebenen Gerätetypen genutzt werden. Insbesondere die Verwendung von `ws4d:AuthenticationEndpoint` zeigt an, dass Geräte die Mechanismen dieses Profils beherrschen.

B.3 Gerätetypen

DPWSec definiert die folgenden Gerätetypen und die damit verbundenen Aufgaben und Eigenschaften.

`ws4d:AuthenticationEndpoint` beschreibt ein Gerät, das Target einer Authentifizierung sein kann. Es MUSS mindestens eine Schnittstelle für eine direkte Authentifizierung mittels Elliptic Curve Diffie Hellman (ECDH) und SOLLTE eine Schnittstelle zur Injektion delegierter Vertrauensbeziehungen haben. Es DARF eine Schnittstelle zur Injektion von Autorisierungen anbieten.

`ws4d:UI-Authenticator` beschreibt ein Gerät, das als Mittelsmann bei einer indirekten Authentifizierung agiert. Es MUSS die selbe Schnittstelle zur direkten ECDH-Authentifizierung anbieten wie ein Endpunkt. Stimmt das Target der angestrebten Authentifizierung nicht mit der eigenen Identifikation überein, MUSS das Gerät eine indirekte Authentifizierung mit dem Target initiieren.

`ws4d:Authenticator` beschreibt ein Gerät, das als Broker bei delegierten Authentifizierungen fungiert. Es MUSS eine Schnittstelle für delegierte Authentifizierungsanfragen bereitstellen.

`ws4d:SynchronousAuthorizer` beschreibt ein Gerät, das als synchroner Autorisierer fungiert. Es MUSS eine Schnittstelle zur Beantragung von Autorisierungstokens anbieten.

`ws4d:AsynchronousAuthorizer` beschreibt ein Gerät, das als asynchroner Autorisierer Nutzerentscheidungen einholt.

B.4 Policies

Durch die folgenden Policies drücken Geräte ihre Fähigkeiten und Konfigurationen aus. Struktur und Details werden an den jeweiligen Stellen beschrieben.

`ws4d:AuthenticationMechanism` definiert OOB-Authentifizierungsmechanismen
`ws4d:SupportedAlgorithms` gibt unterstützte Kombinationen aus Verschlüsselungs- und Signaturalgorithmen sowie KDFs an.
`Brokers` zeigt die derzeit konfigurierten, gültigen Broker an
`Authorizers` zeigt die derzeit konfigurierten, gültigen primären und sekundären Autorisierer an.

B.5 Clients

Clients MÜSSEN identifizierbar sein. Sie sind im Besitz einer URN UUID und diese MUSS den selben Ansprüchen genügen wie denen eines Devices. Sie MÜSSEN eindeutig sein und SOLLTEN einen Neustart des Geräts überstehen.

B.6 Discovery & Description

B.6.1 Unsicheres und sicheres Discovery und Description

Da Discovery und Description der Bildung einer Vertrauensbeziehung vorausgehen können, MÜSSEN diese Vorgänge in diesem Fall als unsicher betrachtet werden. Nach erfolgreicher Authentifizierung MÜSSEN die Vorgänge authentifiziert und unter Sicherstellung der Nachrichtenintegrität wiederholt werden. Einhaltung von Vertraulichkeit ist nicht notwendig. Bei Entstehung von Diskrepanzen zwischen den Angaben vor und nach erfolgter Authentifizierung, MUSS der gesamte Verbindungsaufbau als gescheitert betrachtet und die Vertrauensbeziehung umgehend gelöst werden. Um Signaturen mittels symmetrischer Kryptografie gerecht zu werden, MUSS der verifizierende Client sowohl die Adresse des Ziels als auch seine eigene in die Header-Felder `wsa:To` respektive `wsa:ReplyTo` eintragen.

B.6.2 High- und Low-Effort-Discovery

In Kapitel 6.3.2 wird das Konzept des Low-Effort-Discovery beschrieben. Um es umzusetzen, wird in DPWSec das Headerfeld `<ws4d:loweffort />` definiert. Ist es vorhanden, SOLLTEN nur Geräte antworten, die eine Vertrauensbeziehung

zum Absender der Anfrage unterhalten. Um den Absender der Anfrage identifizieren zu können, trägt er seine Identifikation in das Header-Feld `wsa:ReplyTo` ein.

B.6.2.1 Diskussion

Laut der DPWS-Spezifikation DARF ein Device Probe-Nachrichten ignorieren, wenn im Headerfeld `wsa:ReplyTo` nicht `http://www.w3.org/2005/08/addressing/anonymous` steht. Wenn ein Gerät den hier beschrieben Mechanismus nicht unterstützt, hat dies keine Auswirkungen, da ohnehin nur authentifizierte Geräte gesucht werden, die diesem Profil entsprechen. Bei einem u.U. anschließendem High-Effort-Discovery verhalten sich alle Teilnehmer gemäß der Spezifikation und es kommt zu keinen funktionalen Einbußen.

B.7 Sicherheit auf Nachrichtenebene

B.7.1 WS Compact Security

WS Compact Security bietet ein kompaktes Format zur Einbettung von Signaturen (s. Abschnitt B.7.1.1) und verschlüsselten Nachrichtenpassagen (s. Abschnitt B.7.1.2) in SOAP-Nachrichten. Beispielnachrichten finden sich in Abschnitt B.7.1.3 und in Abschnitt B.7.1.4 ist dargelegt, in welchen Schritten eine Nachricht signiert und/oder verschlüsselt wird.

Die folgenden Überlegungen beziehen sich auf alle Nachrichten außer den Discovery-Verkehr.

Nachrichten SOLLTEN immer signiert werden. Ein Empfänger DARF beschließen, eine nicht signierte Nachricht zu verwerfen. Falls Teile einer Nachricht verschlüsselt werden, SOLLTE die Nachricht signiert werden.

B.7.1.1 Signaturen

Zur Einbettung einer Signatur kommt das WSDD Compact Signature Format zum Einsatz, das in Web Services Dynamic Discovery (WSDD), Version 1.1, in Kapitel 8.2 ([OAS09b, Kap. 8.2]) definiert wird. Als einzige Änderung an diesem Format wird die folgende Konvention eingeführt. Ist das `refs`-Attribut

leer oder fehlt es ganz, so enthält es implizit das gesamte <Envelope>-Element, also die gesamte SOAP-Nachricht.

B.7.1.2 Verschlüsselung

Um Passagen einer SOAP-Nachricht zu verschlüsseln und die Nachricht einzubetten, steht eine Kombination aus dem <enc>-Headerfeld und einem <EncryptedData>-Element im Body der Nachricht zur Verfügung.

Das <enc>-Headerfeld hat das folgende Format:

```
<ws4d:enc
    xmlns:ws4d="http://www.ws4d.org/ws4d-sec/2011/08"
    KeyId=...
    Refs=...
    Scheme=...
/>
```

Listing B.1: enc-Header Feld

KeyId enthält eine Identifikation des verwendeten Schlüssels.
Refs enthält die Liste der verschlüsselten Elemente. Ist dieses Attribut leer oder fehlt es ganz, so erhält es die ID des <Body>-Elements.
Scheme enthält das zur Verschlüsselung eingesetzte Schema. Bisher sind folgende Schemata definiert:

http://www.ws4d.org/ws4d-sec/2011/08/rc4: Verschlüsselung mit dem RC4 Stromchiffrenalgorithmus

http://www.ws4d.org/ws4d-sec/2011/08/rc4-sha1-mac: Signaturbildung: Hash mittels SHA1, Verschlüsselung des Hashes mittels RC4 Stromchiffrenalgorithmus

http://www.ws4d.org/ws4d-sec/2011/08/aes-cbc-128: Verschlüsselung mit dem AES Blockchiffrenalgorithmus

http://www.ws4d.org/ws4d-sec/2011/08/aes-cbc-128-sha1-mac: Signaturbildung: Hash mittels SHA1, Verschlüsselung des Hashes mittels AES Blockchiffrenalgorithmus

Im <Body> der Nachricht können Chiffren mittels <EncryptedData>-Elementen eingebunden werden. Diese sind folgendermaßen aufgebaut:

```
<ws4d:EncryptedData
    xmlns:ws4d="http://www.ws4d.org/ws4d-sec/2011/08"
    wsu:Id=...>
  <ws4d:CipherData>
    ...
  </ws4d:CipherData>
</ws4d:EncryptedData>
```

Listing B.2: Ciphertext im Body der Nachricht

`id` darf fehlen, wenn `header/enc/@refs` leer ist oder fehlt.
`<CipherData>` enthält die Chiffre in base64-codierter Form

B.7.1.3 Beispielnachrichten

Im Folgenden ist eine beispielhafte SOAP-Nachricht in Listing B.3 dargestellt. Listing B.4 zeigt diese Nachricht, die mit dem unter Abschnitt B.7.1.1 beschriebenen Menchanismus signiert und dessen Body mit dem unter Abschnitt B.7.1.2 beschrieben Menchanismus verschlüsselt wurde.

```
<SOAP-ENV:Envelope
    xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Body>
    <ns:single-string-echo xmlns:ns="ws4d:ptest">
      <ns:in>01234</ns:in>
    </ns:single-string-echo>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Listing B.3: Einfacher Request

Der `<Body>` der Nachricht in Listing B.3 enthält genau ein Element mit dem Namen `<single-string-echo>`. Sein Inhalt ist das eine Element `<in>`, das die Zeichenkette `01234` enthält.

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
  <SOAP-ENV:Header>
    <d:Security
        xmlns:d="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01">
      <d:Enc
          KeyId="ZHVtbXk="
          Refs=""
          Scheme="http://www.ws4d.org/ws4d-sec/2011/08/aes">
      </d:Enc>
      <d:Sig
          KeyId="ZHVtbXlkdW1teWR1bW15ZHVtbXk="
          Refs=""
          Scheme="http://www.ws4d.org/ws4d-sec/2011/08/aes-cbc-sha-mac"
          Sig="dtyuCvR713aKt3I4Fv/4bsUQcT7u6JkNvVL0yZFZwE0=">
      </d:Sig>
    </d:Security>
  </SOAP-ENV:Header>
  <SOAP-ENV:Body wsu:id="bodyId">
    <d:EncryptedData
```

```
20          xmlns:d="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01"
21          Id="bodyID">
22        <d:CipherData>
23          ZS40Yxz0H72gN6l00lNbtKNXw/E0Zl Y0YpS2JpEK2q1fkgC... <!-- abgekuerzt -->
24        </d:CipherData>
25      </d:EncryptedData>
26    </SOAP-ENV:Body>
27 </SOAP-ENV:Envelope>
```

Listing B.4: Einfacher Request verschlüsselt und signiert

Zeilen 3 bis 16: `<Security>`-Element.
Zeilen 5 bis 9: `<Enc>`-Element, das die Eigenschaften der verschlüsselten Passagen enthält. Das leere `Refs`-Attribut impliziert, dass nur der Inhalt des `<Body>`-Elements verschlüsselt wurde.
Zeilen 10 bis 15: `<Sig>`-Element, das die Eigenschaften der Signatur und die Signatur selbst enthält. Das leere `Refs`-Attribut impliziert, dass der Hash der gesamten SOAP-Nachricht als Basis der Signatur genutzt wird.
Zeilen 19 bis 25: `<EncryptedData>`-Element.
Zeilen 22 bis 24: `<CipherData>`-Element, das den verschlüsselten Inhalt des Bodys enthält.

B.7.1.4 Vorgehensweise

Im Folgenden werden die Schritte beschrieben, wie die beschriebene Lösung eingesetzt werden muss. Die Reihenfolge der einzelnen Schritte ist verbindlich. Ausgangspunkt ist die in Listing B.3 dargestellte Beispielnachricht.

B.7.1.4.1 Einfügen der Headerfelder In die Ausgangsnachricht wird der `<Security>`-Header hinzugefügt. Er erhält ein leeres `<Sig>`-Unterelement und – falls Teile der Nachricht verschlüsselt werden sollen – ein leeres `<Enc>`-Unterelement. Es ensteht eine Datenstruktur wie unter Listing B.5 abgebildet (beachte Zeilen 2 bis 8)

```
1 <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
2   <SOAP-ENV:Header>
3     <d:Security
4         xmlns:d="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01">
5       <d:Enc></d:Enc>
6       <d:Sig></d:Sig>
7     </d:Security>
8   </SOAP-ENV:Header>
```

```
9   <SOAP-ENV:Body>
10    <ns:single-string-echo xmlns:ns="ws4d:ptest">
11      <ns:in>01234</ns:in>
12    </ns:single-string-echo>
13  </SOAP-ENV:Body>
14 </SOAP-ENV:Envelope>
```

Listing B.5: Ausgangsnachricht mit leeren Header-Feldern

B.7.1.4.2 Attribute festlegen Nach Auswahl der zu verwendenden Algorithmen (Schemes) und Schlüssel (KeyIds), sowie der zu signierenden/verschlüsselnden Nachrichtenteile, werden die entsprechenden Attribute gesetzt. Das Attribut `Security/Sig/@Sig` wird angelegt, bleibt jedoch leer. Es entsteht eine Datenstruktur wie in Listing B.6 dargestellt (s. Zeilen 6-8 und 11-14).

```
1 <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
2   <SOAP-ENV:Header>
3     <d:Security
4         xmlns:d="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01">
5       <d:Enc
6           KeyId="ZHVtbXk="
7           Refs=""
8           Scheme="http://www.ws4d.org/ws4d-sec/2011/08/aes">
9       </d:Enc>
10      <d:Sig
11          KeyId="ZHVtbXlkdW1teWR1bW15ZHVtbXk="
12          Refs=""
13          Scheme="http://www.ws4d.org/ws4d-sec/2011/08/aes-cbc-sha-mac"
14          Sig="">
15      </d:Sig>
16    </d:Security>
17  </SOAP-ENV:Header>
18  <SOAP-ENV:Body>
19    <ns:single-string-echo xmlns:ns="ws4d:ptest">
20      <ns:in>01234</ns:in>
21    </ns:single-string-echo>
22  </SOAP-ENV:Body>
23 </SOAP-ENV:Envelope>
```

Listing B.6: Attribute der Headerfelder angelegt

B.7.1.4.3 Berechnen der Signatur Die Nachricht in Listing B.6 bildet die Grundlage zur Signaturberechnung. Falls das Attribut `Security/Sig/@Refs` leer ist oder fehlt, wird der Digest über die gesamte Nachricht gebildet. Der Verschlüsselte Digest (also die Signatur) wird base64-codiert als Wert des Attri-

buts `Security/Sig/@Sig` festgelegt. Es entsteht die Nachricht wie in Listing B.7 (s. Zeile 15).

```
1 <SOAP-ENV:Envelope
2     xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
3       <SOAP-ENV:Header>
4         <d:Security
5             xmlns:d="http://docs.oasis-open.org/ws-dd/ns/discovery/2009/01">
6           <d:Enc
7               KeyId="ZHVtbXk="
8               Refs=""
9               Scheme="http://www.ws4d.org/ws4d-sec/2011/08/aes">
10          </d:Enc>
11          <d:Sig
12              KeyId="ZHVtbXlkdW1teWR1bW15ZHVtbXk="
13              Refs=""
14              Scheme="http://www.ws4d.org/ws4d-sec/2011/08/aes-cbc-sha-mac"
15              Sig="dtyuCvR713aKt3I4Fv/4bsUQcT7u6JkNvVL0yZFZwE0=">
16          </d:Sig>
17        </d:Security>
18      </SOAP-ENV:Header>
19  <SOAP-ENV:Body>
20    <ns:single-string-echo xmlns:ns="ws4d:ptest">
21      <ns:in>01234</ns:in>
22    </ns:single-string-echo>
23  </SOAP-ENV:Body>
24 </SOAP-ENV:Envelope>
```

Listing B.7: Signatur eingefügt

B.7.1.4.4 Verschlüsselung Falls eine Verschlüsselung durchgeführt wird, erfolgt diese im Anschluss an die Signaturbildung. Der Inhalt des `<Body>`-Elements wird zur Chiffre C verschlüsselt. Anschließend wird der Inhalt des `<Body>`-Elements durch ein `<EncryptedData>`-Element ersetzt. Sein Inhalt ist ein `<CipherData>`-Element, das die Chiffre C in base64-codierter Form enthält. Man erhält die Nachricht wie in Listing B.4 dargestellt.

B.7.2 WS Security Records

Um die Performance zu steigern, indem Signaturbildung und Verschlüsselung durch eine gemeinsame kryptografische Routine bearbeitet werden, existieren die Security Records. Hierbei wird dem `<Body>` ein Element `<Record>` hinzugefügt. Format, Bearbeitung und Namensgebung orientieren sich an dem Record-Protokoll der Transport Layer Security (TLS) Protokollgruppe.

B.7.2.1 Das <Record>-Element

Ein <Record>-Element hat den folgenden Aufbau:

```
<Record
    CipherData="..."
    EncRefs="..."
    KeyId="..."
    PrefixList="..."
    Scheme="..."
    SigRefs="...">
</Record>
```

Listing B.8: Aufbau eines <Record>-Elements

Das <Record>-Element hat keinen Inhalt. Die Informationen sind stattdessen in den Attributen enthalten:
CipherData enthält die verschlüsselten Nachrichtenteile. Siehe dazu den folgenden Abschnitt B.7.2.2.
EncRefs hat die gleiche Bedeutung wie unter Abschnitt B.7.1.2 beschrieben. Darf fehlen oder leer sein. Muss, wenn es nicht leer, ist einzig den Body referenzieren.
KeyId enthält die ID des verwendeten Schlüssels
PrefixList hat die gleiche Bedeutung wie unter WSDD cpt. 8.2 beschrieben
Scheme benennt das verwendete Schema. Bisher sind folgende Schemes definiert:
http://www.ws4d.org/ws4d-sec/2011/08/rc4-sha1-mac: Bildung des Digests mittels SHA-1, Verschlüsselung mittels RC4 Stromchiffre
http://www.ws4d.org/ws4d-sec/2011/08/aes-cbc-128-sha1-mac: Bildung des Digests mittels SHA-1, Verschlüsselung mittels AES CBC 128bit
SigRefs hat die gleiche Bedeutung wie unter Abschnitt B.7.1.1 beschrieben. Darf fehlen oder leer sein.

B.7.2.2 Wert des <CipherData>-Attributs

Um den Wert des CipherData-Attributs zu erhalten, wird die unter Listing B.9 dargestellte Struktur befüllt, mittels des unter Record/Scheme angegebenen Algorithmus verschlüsselt und base64-codiert.

```
<Digest>...</Digest>
<Payload>...</Payload>
```

Listing B.9: Inhalt des <CipherData>-Attributs

<Digest> enthält den base64-codierten Hashwert (=Digest) der zu signierenden Nachrichtenteile.
<Payload> Enthält die zu verschlüsselnden Elemente.

B.7.2.3 Beispielnachricht

Ausgangspunkt ist erneut der einfache Request, der in Listing B.3 abgebildet ist. Werden Vertraulichkeit, Integrität und Authentizität dieser Nachricht mittels eines Security Records mit dem Schema http://www.ws4d.org/ws4d-sec/2011/08/aes-cbc-sha-hmac gewährleistet, entsteht die folgende Nachricht:

```
1 <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/">
2   <SOAP-ENV:Body>
3     <Record
4         CipherData="cLj9mvqgLVL/ty/7wqoaTvXfrlXggP8W[...]" <!-- abgekuerzt -->
5         EncRefs=""
6         KeyId="KeyId"
7         PrefixList=""
8         Scheme="http://www.ws4d.org/ws4d-sec/2011/08/aes-cbc-sha-hmac"
9         SigRefs="">
10    </Record>
11  </SOAP-ENV:Body>
12 </SOAP-ENV:Envelope>
```

Listing B.10: Beispiel eines WS Security Records

B.7.2.4 Vorgehensweise

Die Vorgehensweise zur Ersetzung des Klartextes durch einen Security Record wird im Folgenden beschrieben. Dieses Vorgehen beschreibt die Sendeseite. Auf Seite des Empfängers muss entsprechend in umgekehrter Reihenfolge vorgegangen werden.

B.7.2.4.1 Temporäre Datenstruktur Im Programmspeicher wird eine Datenstruktur *tmp* entsprechend Listing B.9 angelegt.

B.7.2.4.2 Digest der Nachricht Der Digest der zu signierenden Teile der Ausgangsnachricht wird berechnet und in die temporäre Datenstruktur *tmp* eingetragen

B.7.2.4.3 Vertraulichkeit Die zu verschlüsselnden Nachrichtenteile werden in das Element *tmp*/`Payload` verschoben. Sie sind darauf hin nicht mehr in der Ausgangsnachricht enthalten.

B.7.2.4.4 `<Record>`-Element Dem `<Body>` der Ausgangsnachricht wird ein `<Record>`-Element hinzugefügt und die Attribute werden gesetzt. Das Attribut `Record/@Cipherdata` enthält die base64-codierte Chiffre der temporären Datenstruktur *tmp*.

B.7.3 WS Security

Die Festlegung zur Erstellung und Einbettung von Signaturen und Chiffren in SOAP-Envelopes bestreiten zwar den größten Teil der Spezifikation WS Security [OAS06c], es werden jedoch noch weitere Teile geregelt. DPWSec übernimmt aus der originalen Spezifikation die Kapitel bezüglich der ID-Referenzen (4), des Security Headers(5) und der Zeitstempel (10). Kapitel 6 und 7 betreffen Security-Tokens und Referenzen auf diese. Sie werden in DPWSec nicht genutzt, da Security Context Tokens (SCT) implizit durch die KeyIDs von WS Compact Security referenziert werden. Die Kapitel 8 und 9 betreffen die Einbettung von Signaturen und Chiffren und sind aufgrund von WS Compact Security unnötig. Die Kapitel 1-3 und 11-16 sind nicht normativ.

B.8 Sicherheit auf Verbindungsebene

Eine sichere Verbindung wird in der WS Security Spezifikationssuite ebenso wie in DPWSec durch ein SCT repräsentiert. Abschnitt B.9 zeigt seinen Aufbau, und wie es beantragt wird. Das SCT beinhaltet bei Aushändigung eine Identifikation des Kontextes sowie die Verbindungsparameter (zu verwendende Algorithmen und KDF).

Um die Message Freshness von Nachrichten innerhalb einer Verbindung zu gewährleisten und somit Replay-Angriffe zu vermeiden, werden die Message-IDs von WS Addressing (`<wsa:MessageID>`) genutzt, deren Verwendung ohnehin in DPWS vorgesehen ist. In DPWSec erfolgt die Festlegung, dass mit Ausnahme der ersten, jede Message-ID eines Sicherheitskontextes größer sein MUSS, als die vorige (streng monoton steigend). Verliert einer der beiden Teilnehmer die

letzte Message-ID, MUSS die Vertrauensbeziehung aufgegeben und durch eine neue ersetzt werden. Durch die Aushandlung neuen Schlüsselmaterials ist kein Replay-Angriff mehr möglich.

Für zu verwendende abgeleitete Schlüssel (vgl. Kapitel 2.2.2) existiert in DPWSec ein weiterer kompakter Mechanismus. Die zu verwendende KDF wird während des Verbindungsaufbaus festgelegt. Die Verwendung verschiedener KDFs innerhalb einer Verbindung wird in der Literatur nicht gefordert. Der jeweilige Schlüssel wird in den Key-IDs angegeben, indem an die ID des SCT erforderliche Parameter wie Nonce oder Iteration angehängt werden.

B.8.1 WS Secure Conversation

Die Spezifikation WS Secure Conversation [OAS07a] legt das Format eines SCT (Kapitel 2) fest und wie es beantragt (3) und zurückgerufen (6) wird. Das sind die Teile, die in DPWSec übernommen werden. Das Ergänzen eine Kontextes (4) wird nicht benötigt. Der Erneuerung eines Kontextes (5) wird eine vollständige Neuausstellung vorgezogen. Kapitel 7 beschreibt die Einbettung von Angaben abgeleiteter Schlüssel, was in DPWSec durch die o.g. Mechanismen ersetzt wird. Auch die explizite Angabe eines Sicherheitskontextes (8) wird durch die KeyIDs ersetzt und nicht in DPWSec übernommen.

B.9 Authentifizierung

B.9.1 Grundlegendes

Während einer aktiven indirekten Authentifizierung SOLLTE ein Gerät nicht auf eine Suchanfrage nach UI-Authentifizierern antworten und SOLLTE weitere eintreffende Requests mit einem Fehler quittieren. Dies dient der Vermeidung von Schleifen.

Clients SOLLTEN bis zum Erfolgsfall jeden verfügbaren UI-Authentifizierer durchiterieren. UI-Authentifizierer MÜSSEN bei Kaskadierung jeden UI-Authentifizierer durchiterieren. Das dient der Auflösung falscher Abzweigungen.

B.9.2 Policies

B.9.2.1 Authentifizierungsmechanismen

Dient der Angabe unterstützter Authentifizierungsmechanismen.
Policy-Name: `ws4d:AuthenticationMechanism`

B.9.2.1.1 Aufbau Es können beliebig viele Authentifizierungsmechanismen angegeben werden.

```
1 (<wsp:Policy Name="AuthenticationMechanism"
2              xmlns:wsp="http://www.w3.org/ns/ws-policy">
3   <wsp:ExactlyOne>
4     (<wsp:All>
5        <!-- Mechanismus -->
6      </wsp:All>)*
7   </wsp:ExactlyOne>
8 </wsp:Policy>)?
```

Listing B.11: Struktur der AuthenticationMechanism-Policies

B.9.2.1.2 Definierte Authentifizierungsmechanismen In dieser Arbeit sind die folgenden Mechanismen definiert und umgesetzt worden. Streng genommen handelt es sich nicht um Teile des Profils DPWSec, sondern um anwendungsbezogene Ergänzungen.

`<ws4d:FlickerAuthentication>`, `<ws4d:TappingAuthentication>` und `<ws4d:PinAuthentication>` bezeichnen die jeweiligen OOB-Authentifizierungsmechanismen für das Flackern einer Pin, das Eintastern respektive der numerischen Ein- oder Ausgabe.

`<ws4d:BrokeredAuthentication>` und `ws4d:EncryptedPinExchange` sind Spezialfälle. Ersteres gibt an, dass der Endpunkt delegierte Authentifizierung unterstützt, letzteres dient der Protokollerweiterung des Encrypted Pin Exchange (vgl. Kapitel 4.5.2).

B.9.2.2 Broker

Dient der Angabe akzeptierter Broker.
Policy-Name: `ws4d:Brokers`

B.9.2.2.1 Aufbau Es können beliebig viele Broker angegeben werden.

```
<wsp:Policy Name="Brokers" xmlns:wsp="http://www.w3.org/ns/ws-policy">
  <wsp:ExactlyOne>
    (<wsp:All>
      <ws4d:broker address="[Broker-Address]" />
    </wsp:All>)*
  </wsp:ExactlyOne>
</wsp:Policy>
```

Listing B.12: Struktur der BrokersPolicies

B.9.3 Port-Types & Input Actions

Das lokale Ressourcen-Discovery wird mithilfe von Port-Types des STS und Input-Actions der Methoden durchgeführt.

B.9.3.1 Dienst-Port-Types

Der STS eines Endpunkts, Authentifizierers oder UI-Authentifizierers lässt sich daran erkennen, dass er das Issue-Binding, das Validate-Bindung oder beide anbietet.

`wst:Issue` – zeigt an, dass ein SCT beantragt werden kann. Direkt, delegiert oder beides, je nach Gerätetyp.

`wst:Validate` – zeigt an, dass ein SCT eingespielt werden kann. Nur für Endpunkte.

B.9.3.1.1 Elliptic Curve Diffie Hellman (ECDH) Bietet ein STS die direkte Authentifizierung mittels ECDH an, lässt sich das über die Input- und Output-Actions der beiden angebotenen Operationen anzeigen. Operationen lassen sich über diese Actions filtern.

Request1: `http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue`

Response1: `http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTR/Issue`

Request1: `http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/Issue`

Response2: `http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTRC/IssueFinal`

Die Authentifizierungsparameter Die während eines ECDH-Handshakes ausgetauschten kryptografischen Parameter haben die folgende Form:

```
<ws4d:AuthECCDHParameters>
  <ws4d:auth-ecc-dh-curvename>
  <ws4d:auth-ecc-dh-nonce></ws4d:auth-ecc-dh-nonce>
  <ws4d:auth-ecc-dh-public-key></ws4d:auth-ecc-dh-public-key>
  <ws4d:auth-ecc-dh-cmac></ws4d:auth-ecc-dh-cmac>
  <ws4d:auth-ecc-dh-numerical-pin></ws4d:auth-ecc-dh-numerical-pin>
</ws4d:AuthECCDHParameters>
```

Listing B.13: AuthECCDHParameters

Wird die numerische Pin in `<ws4d:auth-ecc-dh-numerical-pin>` zum Zweck des Encrypted Pin Exchange einfügt, MUSS die Nachricht verschlüsselt und signiert übertragen werden. Eine Nachricht, die dieser Anforderung nicht genügt DARF NICHT verarbeitet oder weitergeleitet werden.

Das Requested Security Token und Security Context Token Die Auslieferung eins SCT erfolgt laut WS Secure Conversation und WS Trust wie folgt:

```
<wst:RequestedSecurityToken>
  <wsc:SecurityContextToken>
    <wsc:Identifier>
    <wst:SignWith>
    <wst:EncryptWith>
  </wsc:SecurityContextToken>
</wst:RequestedSecurityToken>
```

Listing B.14: Requested Security Token

ECC_DH1 Methode für erste Hälfte des Handshakes. Das Requested Security Token ist optional bereits enthalten, damit Verbindungsparameter (Algorithmen, Kurvennamen, ...) beantragt werden können

Eingabe:

```
<wst:RequestSecurityToken>
  <wst:TokenType></wst:TokenType>
  <wst:RequestType></wst:RequestType>
  <wsp:AppliesTo></wsp:AppliesTo>
  <wst:OnBehalfOf></wst:OnBehalfOf>
  <ws4d:AuthenticationType></ws4d:AuthenticationType>
  <wst:SignWith>
  <wst:EncryptWith>
  <ws4d:Brokerable></ws4d:Brokerable>
  <ws4d:AuthECCDHParameters>
    [...]
```

```
12   </ws4d:AuthECCDHParameters>
13 </wst:RequestSecurityToken>
```

Listing B.15: Struktur eines ECC_DH1-Requests

Ausgabe:

```
1 <wst:RequestSecurityTokenResponse>
2   <wst:TokenType></wst:TokenType>
3   <wst:RequestType></wst:RequestType>
4   <wsp:AppliesTo></wsp:AppliesTo>
5   <wst:OnBehalfOf></wst:OnBehalfOf>
6   <ws4d:AuthenticationType></ws4d:AuthenticationType>
7   <ws4d:AuthECCDHParameters></ws4d:AuthECCDHParameters>
8   <wst:RequestedSecurityToken>
9     [...]
10  </wst:RequestedSecurityToken>
11 </wst:RequestSecurityTokenResponse>
```

Listing B.16: Struktur eines ECC_DH1-Response

ECC_DH2 Zweite Hälfte

In:

```
1 <wst:RequestSecurityTokenResponse>
2   [...] <!-- s.o. -->
3 </wst:RequestSecurityTokenResponse>
```

Listing B.17: Struktur eines ECC_DH2-Request

Out:

```
1 <wst:RequestSecurityTokenResponseCollection>
2   <wst:RequestSecurityTokenResponse>
3     [...] <!-- s.o. -->
4   </wst:RequestSecurityTokenResponse>
5 </wst:RequestSecurityTokenResponseCollection>
```

Listing B.18: Struktur eines ECC_DH2-Response

B.9.3.1.2 Delegierte Authentifizierung Die Methode zur Beantragung einer delegierten Authentifizierung zeichnet sich durch folgende Input- und Output-Actions aus:

Request: `http://docs.oasis-open.org/ws-sx/ws-trust/200512/RST/SCT`

Response: `http://docs.oasis-open.org/ws-sx/ws-trust/200512/RSTR/SCT`

Das `RequestedSecurityTokenResponse` in der Antwort enthält ein `RequestedSecurityToken` genau wie für ECDH, jedoch zusätzlich ein `wst:RequestedProofToken`, in dem der Schlüssel als `wst:BinarySecret` abgelegt ist.

Komplette Antwort:

```
<wst:RequestSecurityTokenResponseCollection>
  <wst:RequestSecurityTokenResponse>
    <wst:RequestedSecurityToken>
      <wsc:SecurityContextToken>
      </wsc:SecurityContextToken>
    </wst:RequestedSecurityToken>
    <wst:RequestedProofToken>
      <wst:BinarySecret
          type=http://docs.oasis-open.org/ws-sx/ws-trust/200512/SymmetricKey>
      <!-- s. wst,cpt3.3 -->
    </wst:RequestedProofToken>
  </wst:RequestSecurityTokenResponse>
</wst:RequestSecurityTokenResponseCollection>
```

Listing B.19: RSTRC: Antwort auf RST

B.9.4 Aufhebung von Vertrauensverhältnissen

Jedes Gerät MUSS das Cancel-Binding implementieren (WS Secure Conversation, Kapitel 6), damit Vertrauensverhältnisse aufgehoben werden können. Jeder Client SOLLTE die Eventquelle seines primären Brokers abonnieren, über die Vertrauensbeziehungen widerrufen werden können. Dieser Mechanismus wird als Federated-Sign-Out bezeichnet und ist in WS Federation [L+06, Kapitel 4.2] festgelegt.

Ein Client KANN über das Cancel-Binding direkt einzelne Vertrauensverhältnisse auflösen oder KANN über eine Signout-Message an seinen STS komplett austreten. Dazu richtet der Client eine Sign-Out-Nachricht an seinen STS (vgl. [L+06, Kapitel 4.1]), der in seinem Namen die Cancel-Bindings der Geräte ansprechen MUSS, mit denen der Client ein Vertrauensverhältnis unterhält.

Durch eine zentrale Konfiguration SOLLTE ein STS ebenfalls angewiesen werden können, alle Vertrauensbeziehungen eines Geräts aufzulösen. Devices erfahren dies über ihr Cancel-Binding, Clients über die Event-Quelle. STSes MÜSSEN alle Kontexte (auch kaskadiert delegierte) speichern, um Cancel-Nachrichten verschicken zu können, die vom ausgetretenen Gerät signiert wurden.

B.9.5 WS Trust und WS Federation

Das von WS Trust ([OAS06g]) eingeführte Konzept des Security Token Service (STS) ist ein zentrales Element von DPWSec und wird gemäß dem nicht normativen Kapitel 2 übernommen. Kapitel 3 definiert die Nachrichtenformate für die Beantragung und Auslieferung von Security Tokens und wie binäre Shared Secrets in eine Nachricht eingebettet werden. Diese Teile werden uneingeschränkt in DPWSec übernommen.

Die Kapitel 4, 5 und 6 legen fest, wie Security Tokens ausgestellt, erneuert und widerrufen werden. Da dies für SCTs bereits in WS Secure Conversation festgelegt ist und DPWSec keine alternativen Tokenformen benötigt, werden Kapitel 5 und 6 vollständig verworfen. Aus Kapitel 4 werden ausschließlich die Nachrichtenformate zur Beantragung und Auslieferung (4.1-4.4) übernommen. Abschnitt 4.4 wird jedoch dahingehend eingeschränkt, dass nur die Einbettung eines unverschlüsselten binären Shared Secret in das auszuliefernde PoP-Token berücksichtigt wird. Dies ist zulässig, da die gesamte Nachricht verschlüsselt übertragen wird.

Die in Kapitel 7 definierte Validate-Schnittstelle wird vollständig übernommen. Es sei jedoch anzumerken, dass die Semantik stark verändert ist. So dient sie nicht etwa einer Validierung von SCTs (oder später Autorisierungstokens) sondern dem asynchronen Einspielen dieser Tokens bei der delegierten Authentifizierung oder synchronen Autorisierung (vgl. Abbildungen 6.11 auf Seite 139 bzw. 6.21 auf Seite 155).

Kapitel 8 definiert weiterhin ein Framework zur Aushandlung zusätzlicher Verbindungsparameter. Diese Möglichkeit ist in DPWSec nicht vorgesehen. Es wird Unterkapitel 8.1 berücksichtigt, da es festlegt, wie mehrstufige Handshakes ausgeführt werden müssen. Dies kommt beim ECDH-Handshake zum Tragen. Kapitel 9 definiert eine Reihe von Erweiterungen für Anfragen. Die einzige, die übernommen wird, ist der `wst:OnBehalfOf`-Parameter aus Unterkapitel 9.1. Er wird genutzt, um bei delegierten, kaskadiert delegierten und indirekten Authentifizierungen den eigentlichen Requestor zu benennen. Das Key-Exchange-Token aus Kapitel 10 findet keine Verwendung.

Schließlich wird aus WS Federation ([L+06]) das gesamte Kapitel 4 übernommen. Es beschreibt clientseitige Sign-Out-Nachrichten und den Federated-Sign-Out. Diese beiden Konzepte wurden weiter oben erläutert.

B.10 Autorisierung

B.10.1 Policies

B.10.1.1 Angabe der zuständigen Autorisierer

Dient der Angabe konfigurierter synchroner Autorisierer.
Policy-Name: `ws4d:Authorizer`

B.10.1.1.1 Aufbau Eine Authorizer-Policy ist wie folgt aufgebaut:

```
<wsp:Policy Name="Authorizers" xmlns:wsp="http://www.w3.org/ns/ws-policy">
  <wsp:ExactlyOne>
    <wsp:All>
      <ns:PrimaryAuthorizer address="[Address1]" />?
      <ns:Authorizer address="[Address2]" />*
    </wsp:All>
  </wsp:ExactlyOne>
</wsp:Policy>
```

Listing B.20: Aufbau einer Authorizer-Policy

B.10.2 Anfragen einer Autorisierung

Eine Autorisierung wird bei einem STS beantragt. Die Unterscheidung zur Beantragung einer Authentifizierung liegt in der Angabe des unterschiedlichen `wst:TokenType`-Elements. Die Beantragung einer Autorisierung folgt dem folgenden Format:

```
<wst:RequestSecurityToken xmlns:wst="...">
  <wst:TokenType>
    urn:oasis:names:tc:SAML:1.0:assertion
  </wst:TokenType>
  <wst:RequestType>
    http://docs.oasis-open.org/ws-sx/ws-trust/200512/Issue
  </wst:RequestType>
  <wsp:AppliesTo><!-- Adresse des Subjekts --></wsp:AppliesTo>
  (<wst:Claims
Dialect="http://docs.oasis-open.org/wsfed/authorization/200706/authclaims">
    <auth:Claim> <!-- s.u. --> </auth:Claim>+
  </wst:Claims>)?
  (<auth:AdditionalContext>
    <auth:ContextItem> <!-- s.u. --> </auth:ContextItem>+
  </auth:AdditionalContext>)?
</wst:RequestSecurityToken xmlns:wst="...">
```

Listing B.21: Beantragen einer Autorisierung

Das Format von Claims wird nachfolgend dargestellt. Der in WS Federation definierte Claimdialekt `http://docs.oasis-open.org/wsfed/authorization/200706/authclaims` SOLLTE aus Interoperabilitätsgründen genutzt werden. `auth:Claim` verlangt ein Attribut `URI`, das per Default auf die `wsa:Action` verweist. Aufgrund der Redundanz wird dies in DPWSec als optional festgelegt und verweist bei fehlen implizit auf die `wsa:Action`.

```
<auth:Claim>
  <auth:DisplayName ...> xs:string </auth:DisplayName>?
  <auth:Description ...> xs:string </auth:Description>?
  <auth:DisplayValue ...> xs:string </auth:DisplaValue>?
  (<auth:Value>...</auth:Value> |
   <auth:StructuredValue ...>...</auth:StructuredValue> |
   (<auth:EncryptedValue @DecryptionCondition="xs:anyURI">
      <xenc:EncryptedData>...</xenc:EncryptedData>
    <auth:EncryptedValue>) |
   <auth:ConstrainedValue>...</auth:ConstrainedValue>
  )?
</auth:Claim>
```

Listing B.22: Format eines Claims

`auth:ConstrainedValue` kann kleiner als ein Wert, größer als ein Wert oder beides sein. An dieser Stelle sei auf WS Federation, Kapitel 9.3.1 verwiesen.

```
<auth:ContextItem Name="NameDesKontextElements">
  <auth:Value>WertDesElements</authValue>
</auth:ContextItem>
```

Listing B.23: Format einer zusätzlichen Kontextinformation

B.10.3 Auslieferung einer Autorisierung an Anfragesteller

Ebenso, wie die Anfrage eines Autorisierungstokens der einer Authentifizierung ähnelt, gilt gleiches für die Auslieferung. Sie erfolgt durch eine

`<RequestSecurityTokenResponseCollection>`.

B.10.4 Format einer Autorisierungsentscheidung

Es folgt das Format eines solchen Autorisierungstokens. Es entspricht vollständig der in SAML ([CKPM05]) definierten Form, mit der Ausnahme, das anstelle einer klassischen XML-Signature eine WSDD Compact Signature genutzt wird.

```
<saml:Assertion>
  <saml:Version></saml:Version>
  <saml:ID></saml:ID>
  <saml:IssueInstant></saml:IssueInstant>
  <saml:Issuer></saml:Issuer>
  <wsdd:Sig></wsdd:Sig>?
  <saml:Subject>
  <saml:Conditions>?
  <saml:AuthzDecisionStatement Resource="..." Decision="...">
    <saml:Action Namespace="..."></saml:Action>+
    <saml:Evidence>
      <saml:Assertion></saml:Assertion>+
    </saml:Evidence>?
  </saml:AuthzDecisionStatement>
</saml:Assertion>
```

Listing B.24: Format einer Autorisierungsentscheidung

Die eigentliche Entscheidung – der Inhalt des `<Decision>`-Elements kann die Werte Permit, Deny oder Indeterminate annehmen. Der (implizite) Default für die erlaubten Actions ist das Schema `urn:oasis:names:tc:SAML:1.0:action:rwedc-negation`. Für DPWSec wird als Default das kompaktere und ausreichende Schema `urn:oasis:names:tc:SAML:1.0:action:ghpp` (Kapitel 8.1.3 in [CKPM05]) angenommen.

B.10.5 Sekundäre Autorisierer

Als spezielle Ressource für die Delegation bzw. Festlegung sekundärer Autorisierer wird die Ressource `ws4d:secondAuth` definiert. So kann bei Beantragung oder Validierung deutlich gemacht werden, dass es sich um einen Antrag handelt, sekundärer Autorisierer des Subjekts zu werden.

B.10.6 Fehler

Durch das Konzept der asynchronen und der sekundären Autorisierer entstehen mögliche Fehlerfälle, die nicht durch die Basisspezifikationen abgedeckt werden. Daher werden die folgenden Fehlerfälle und- codes definiert:

Fehler-String	Fehler-Code
The supplied user credential is wrong or can not be verified.	`ws4d:WrongCredential`
The subject will not accept the requestor as secondary authorizer	`ws4d:NoSecondaryAuthorizersAccepted`

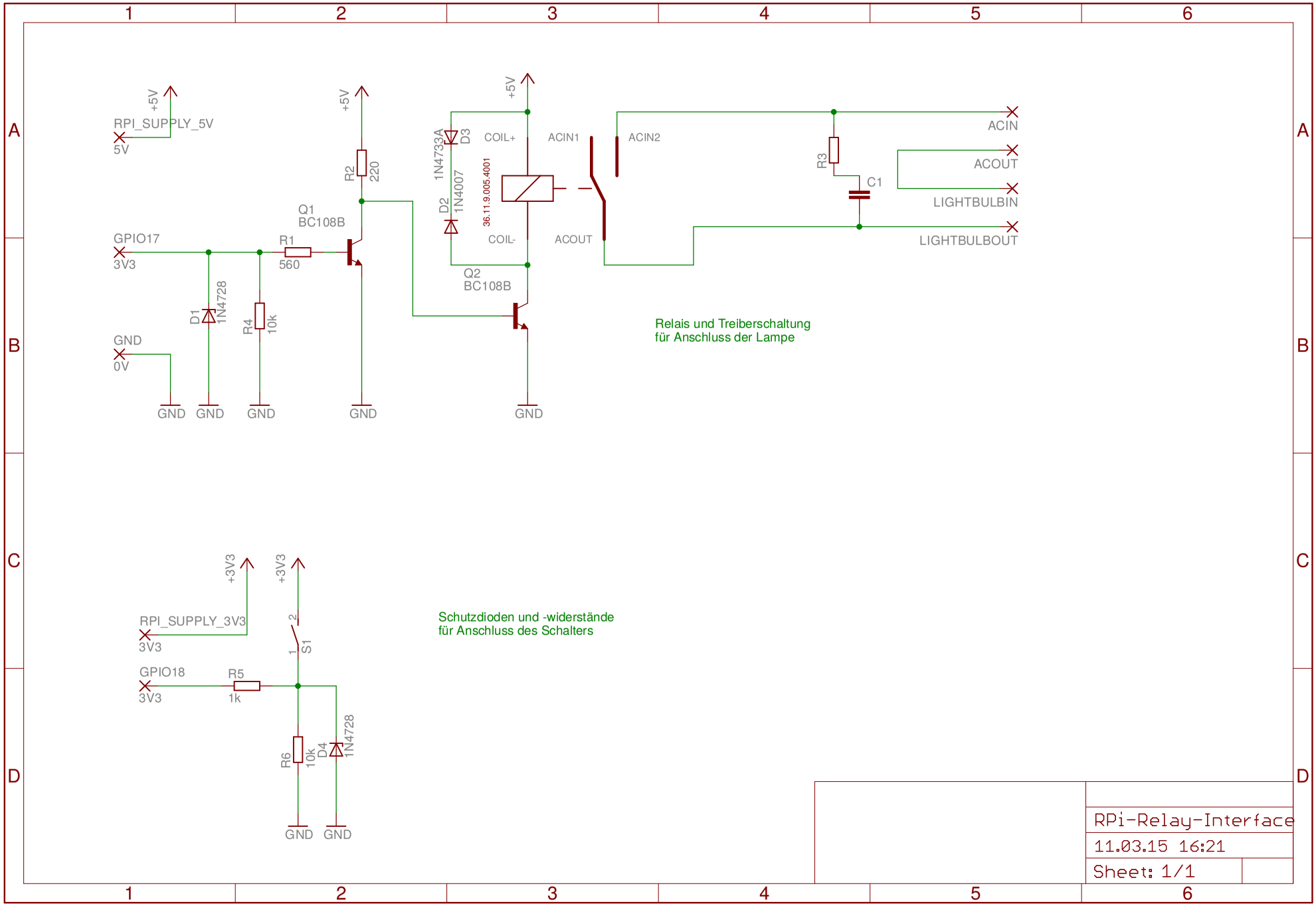
Tabelle B.2: Definition zusätzlicher Fehlertypen

B.10.7 WS Federation und SAML

Die Spezifikationen WS Federation [L+06] und SAMLCore [CKPM05] regeln Autorisierung umfassend. Ersteres beschreibt Beantragung und Auslieferung einer Autorisierungsentscheidung, letztere, wie eine Autorisierungsentscheidung aufgebaut ist. Die entsprechenden Teile werden vollständig und ohne Einschränkung übernommen, um die Flexibilität der Autorisierungsinfrastruktur nicht einzuschränken. Es handelt sich dabei um das gesamte Kapitel 9 von WS Federation und die Abschnitte 2.1-2.6, 2.7.4 und 8.1 der SAML-Core-Spezifikation, die Aufbau und eventuelle Einschränkungen einer Autorisierungsentscheidung beschreiben.

Anhang C

Schaltpläne


RPI_SUPPLY_5V
5V
+5V
GPIO17
3V3
R1
560
Q1
BC108B
R2
220
D1
1N4728
R4
10k
GND
0V
GND
GND
GND
GND
+5V
1N4733A
D3
COIL+
ACIN1
ACIN2
D2
1N4007
36.11.9.005.4001
COIL-
ACOUT
Q2
BC108B
GND
R3
C1
ACIN
ACOUT
LIGHTBULBIN
LIGHTBULBOUT
Relais und Treiberschaltung
für Anschluss der Lampe
RPI_SUPPLY_3V3
3V3
+3V3
+3V3
S1
GPIO18
3V3
R5
1k
R6
10k
D4
1N4728
GND
GND
Schutzdioden und -widerstände
für Anschluss des Schalters
RPi-Relay-Interface
11.03.15 16:21
Sheet: 1/1

Literaturverzeichnis

[A+15] Alewa, Christos u. a.: [TLS] TLS 1.3 - Support for compression to be removed – Diskussion auf der IETF-Mailing-Liste zum Thema TLS. Online: https://www.ietf.org/mail-archive/web/tls/current/msg17716.html (Zuletzt geprüft: 29.09.2015), September 2015

[AB09] Adetoye, Adedayo O. ; Badii, Atta: Foundations and Applications of Security Analysis. Berlin, Heidelberg : Springer-Verlag, 2009. – ISBN 978–3–642–03458–9, Kapitel A Policy Model for Secure Information Flow, S. 1–17

[ABCR10] Auletta, Vincenzo ; Blundo, Carlo ; Cristofaro, Emiliano D. ; Raimato, Guerriero: Web Services Invocation over Bluetooth. In: Wireless Sensor Network 2 (2010), Nr. 6, S. 447–461. http://dx.doi.org/10.4236/wsn.2010.26056. – DOI 10.4236/wsn.2010.26056

[AGR+05] Ahler, Marco ; Grinewitschus, Viktor ; Ressel, Christian ; Miranda, José M. ; Ramos Álvaro ; Mevissen, Ron ; Tobies, Stephan: Detailed Design of the Amigo Middleware Core Security & Privacy , Content Distribution, Data Storage / IST Amigo Project. 2005. – Forschungsbericht

[AKM07] Aitenbichler, Erwin ; Kangasharju, Jussi ; Mühlhäuser, Max: MundoCore: A Light-weight Infrastructure for Pervasive Computing. In: Pervasive and Mobile Computing (2007), S. 332–361. http://dx.doi.org/doi:10.1016/j.pmcj.2007.04.002. – DOI doi:10.1016/j.pmcj.2007.04.002

[All06] Allegro: Networked Digital Media Standards – A UPnP / DLNA Overview / Allegro Software Development Corporation. 2006. – Forschungsbericht

[AMCK+02] Al-Muhtadi, J. ; Campbell, R. ; Kapadia, A. ; Mickunas, M.D. ; Yi, Seung: Routing through the mist: privacy preserving communication in ubiquitous computing environments. In: Distributed Computing Systems, 2002. Proceedings. 22nd International Conference on, 2002. – ISSN 1063–6927, S. 74–83

[AMRCM03] Al-Muhtadi, J. ; Ranganathan, A. ; Campbell, R. ; Mickunas, M.D.: Cerberus: a context-aware security scheme for smart spaces. In: Pervasive Computing and Communications, 2003. (PerCom 2003). Proceedings of the First IEEE International Conference on, 2003, S. 489 – 496

[AS05] Alshamsi, AbdelNasir ; Saito, Takamichi: A Technical Comparison of IPSec and SSL. In: Proceedings of the 19th International Conference on Advanced Information Networking and Applications - Volume 2. Washington, DC, USA : IEEE Computer Society, 2005 (AINA '05). – ISBN 0–7695–2249–1, S. 395–398

[BB+06] Ballinger, K. ; Bisset, B. u. a.: Web Services Metadata Exchange (WS-MetadataExchange) Version 1.1. Online: http://specs.xmlsoap.org/ws/2004/09/mex/WS-MetadataExchange.pdf (Zuletzt geprüft: 29.09.2015), August 2006

[BBB+11] Barker, Elaine ; Barker, William ; Burr, William ; Polk, William ; Smid, Miles: Recommendation for Key Management – Part 1: General (Revision 3) / National Institute of Standards and Technology (NIST). 2011 (SP 800-57). – Forschungsbericht

[BBF+08] Bartel, Mark ; Boyer, John ; Fox, Barb ; LaMacchia, Brian ; Simon, Ed: XML Signature Syntax and Processing (Second Edition) – W3C Recommendation. June 2008

[BCC+07] Ben Mokhtar, Sonia ; Caporuscio, Mauro ; Charlet, Damien ; Fredj, Manel ; Georgantas, Nikolaos ; Issarny, Valerie ; Raverdy, Pierre-Guillaume ; Zahn, Thomas ; Benesch, Mathias ; Berger, Frank ; Eikerling, Heinz-Josef ; Mascolo, Cecilia ; Namara, Liam M. ; Carzaniga, Antonia ; Hall, Cyrus ; Rose, Jeff ; Wolf, Alexander L. ; Toulis, Panos ; Manifavas, Harry ; Liotopoulos, Fotis K.: Deliverable D3.1 – Middleware Specification and Architecture. 2007

[BCQ+10] Baldoni, Roberto ; Cerocchi, Adriano ; Querzoni, Leonardo ; Rasch, Katharina ; Li, Fei ; Dustdar, Schahram ; Aiello, Marco ; Khoury, Elie el ; Doğanay, Mahir C. ; Rumiz, Antonio ; Pucci, Paolo ; García, Fernando O. ; Rafael, Pablo A. ; Milagro, Francisco: Deliverable 6.2.a (Version 1) Adaptive middleware services. 2010

[BDCM+09] Baldoni, R. ; Di Ciccio, C. ; Mecella, M. ; Patrizi, F. ; Querzoni, L. ; Santucci, G. ; Dustdar, S. ; Li, F. ; Truong, H.-L. ; Albornos, L. ; Milagro, F. ; Rafael, P.A. ; Ayani, R. ; Rasch, K. ; Lozano, M.G. ; Aiello, M. ; Lazovik, A. ; Denaro, A. ; Lasala, G. ; Pucci, P. ; Holzner, C. ; Cincotti, F. ; Aloise, F.: An Embedded Middleware Platform for Pervasive and Immersive Environments for-All. In: Sensor, Mesh and Ad Hoc Communications and Networks Workshops, 2009. SECON Workshops '09. 6th Annual IEEE Communications Society Conference on, 2009

[BER02] Boyer, John ; Eastlake, Donald E. ; Reagle, Joseph: Exclusive XML Canonicalization Version 1.0. 2002

[Beu09] Beutelspacher, Albrecht: Kryptologie – Eine Einführung in die Wissenschaft vom Verschlüsseln, Verbergen und Verheimlichen. 9. Auflage. Vieweg + Teubner Verlag, 2009. – ISBN 978–3–8348–0703–8

[BGI07] Ben Mokhtar, Sonia ; Georgantas, Nikolaos ; Issarny, Valérie: COCOA: COnversation-based service COmposition

in pervAsive computing environments with QoS support. In: Journal of Systems and Software 80 (2007), Nr. 12, S. 1941–1955

[BGS+06] Broxval, M. ; Gritti, M. ; Saffiotti, A. ; Seo, B.S. ; Cho, Y.J.: PEIS Ecology: Integrating Robots into Smart Environments. In: IEEE Int. Conf. on Robotics and Automation (ICRA). Orlando, USA, Mai 2006

[BJH+07] Bai, Yuebin ; Ji, Haixing ; Han, Qingmian ; Huang, Jun ; Qian, Depei: MidCASE : A Service Oriented Middleware Enabling Context Awareness for Smart Environment. In: Multimedia and Ubiquitous Engineering, 2007. MUE '07. International Conference on, 2007, S. 946–951

[BK01] Barton, John ; Kindberg, Tim: The Cooltown User Experience / Hewlett Packard Laboratories Palo Alto. 2001. – Forschungsbericht

[BK12] Barker, Elaine ; Kelsey, John: Recommendation for Random Number Generation Using Deterministic Random Bit Generators. 2012

[BKS+08] Kapitel 14. In: Bagci, Faruk ; Kluge, Florian ; Satzger, Benjamin ; Pietzowski, Andreas ; Trumler, Wolfgang ; Ungerer, Theo: Experiences with a Smart Office Project. John Wiley & Sons, Inc, 2008. – ISBN 978–0470195550, S. 294–319

[Blu07] Bluetooth SIG: Specification of the Bluetooth System (2.1 + EDR). 2007

[Blu10a] Bluetooth SIG: Specification of the Bluetooth System (4.0). 2010

[Blu10b] Bluetooth SIG: Vortragsfolien: Bluetooth Low Energy Technology Training. 2010

[BMC+05] Bless, Roland ; Mink, Stefan ; Conrad, Michael ; Kutzner, Kendy ; Blass, Erik-Oliver ; Hof, Hans-Joachim ; Schöller, Marcus: Sichere Netzwerkkommunikation – Grundlagen,

Protokolle und Architekturen. 2005. http://dx.doi.org/10.1007/3-540-27896-6. http://dx.doi.org/10.1007/3-540-27896-6. – ISBN 978–3–540–27896–2

[BNS10] Beutelspacher, Albrecht ; Neumann, Heike B. ; Schwarzpaul, Thomas: Kryptografie in Theorie und Praxis. 2. Auflage. Springer Vieweg Verlag, 2010. – ISBN 978–3–8348–9631–5

[Bra97] Bradner, S.: RFC 2119: Key words for use in RFCs to Indicate Requirement Levels. Version: 1997. https://www.ietf.org/rfc/rfc2119.txt

[BRBS07] Bordignon, M. ; Rashid, J. ; Broxval, M. ; Saffiotti, A.: Seamless Integration of Robots and Tiny Embedded Devices in a PEIS-Ecology. In: IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS). San Diego, USA, 2007

[BSGR03] Becker, C. ; Schiele, G. ; Gubbels, H. ; Rothermel, K.: BASE - a micro-broker-based middleware for pervasive computing. In: Pervasive Computing and Communications, 2003. (PerCom 2003). Proceedings of the First IEEE International Conference on, 2003, S. 443–451

[Bun09] Bundesamt für Sicherheit in der Infromationstechnik: SOA-Security-Kompendium – Sicherheits in Service-orientierten Architekturen – Version 2.0. 2009

[CAMCM05] Chetan, Shiva ; Al-Muhtadi, Jalal ; Campbell, Roy ; Mickunas, M. D.: Mobile Gaia: a middleware for ad-hoc pervasive computing. In: Consumer Communications and Networking Conference, 2005. CCNC. 2005 Second IEEE, 2005, S. 223–228

[CAMN+02] Campbell, Roy ; Al-Muhtadi, Jalal ; Naldurg, Prasad ; Sampemane, Geetanjali ; Mickunas, M. D.: Towards security and privacy for pervasive computing. In: Proceedings of the 2002 Mext-NSF-JSPS international conference on Software security: theories and systems. Berlin, Heidelberg : Springer-Verlag, 2002 (ISSS'02). – ISBN 3–540–00708–3, 1–15

[CBB+12] Conzon, D. ; Bolognesi, T. ; Brizzi, P. ; Lotito, A. ; Tomasi, R. ; Spirito, M.A.: The VIRTUS Middleware: An XMPP Based Architecture for Secure IoT Communications. In: Computer Communications and Networks (ICCCN), 2012 21st International Conference on, 2012, S. 1–6

[CC03] Chan, Alvin T. S. ; Chuang, Siu-Nam: MobiPADS: A Reflective Middleware for Context-Aware Mobile Computing. In: IEEE Trans. Softw. Eng. 29 (2003), Dezember, Nr. 12, 1072–1085. http://dx.doi.org/10.1109/TSE.2003.1265522. – DOI 10.1109/TSE.2003.1265522. – ISSN 0098–5589

[CFZA02] Covington, M.J. ; Fogla, P. ; Zhan, Zhiyuan ; Ahamad, M.: A context-aware security architecture for emerging applications. In: Computer Security Applications Conference, 2002. Proceedings. 18th Annual, 2002. – ISSN 1063–9527, S. 249 – 258

[Che09] Chen, Lily: Recommendation for Key Derivation Using Pseudorandom Functions (Revised). 2009

[CKPM05] Cantor, Scott ; Kemp, John ; Philpott, Rob ; Maler, Eve: Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0. Online: http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf (Zuletzt geprüft: 29.09.2015), März 2005

[Cor12] Coronato, Antonio: Uranus: A Middleware Architecture for Dependable AAL and Vital Signs Monitoring Applications. In: Sensors 12 (2012), Nr. 3, S. 3145–3161. http://dx.doi.org/10.3390/s120303145. – DOI 10.3390/s120303145. – ISSN 1424–8220

[CPC+04] Chen, Harry ; Perich, Filip ; Chakraborty, Dipanjan ; Finin, Tim ; Joshi, Anupam: Intelligent Agents Meet Semantic Web in a Smart Meeting Room. In: Proceedings of the Third International Joint Conference on Autonomous Agents

and Multiagent Systems - Volume 2. Washington, DC, USA : IEEE Computer Society, 2004 (AAMAS '04). – ISBN 1–58113–864–4, 854–861

[CRI07] Cardoso, Roberto S. ; Raverdy, Pierre-Guillaume ; Issarny, Valérie: A Privacy-Aware Service Discovery Middleware for Pervasive Environments. In: Trust Management Bd. 238. Springer US, 2007, S. 59–74

[CRI10] Caporuscio, Mauro ; Raverdy, Pierre-Guillaume ; Issarny, Valérie: ubiSOAP: A Service Oriented Middleware for Ubiquitous Networking. In: IEEE Transactions on Services Computing (2010)

[DA99] Dierks, T. ; Allen, C.: RFC 2246: The TLS Protocol Version 1.0. January 1999

[DAP+12] Danakis, C. ; Afgani, M. ; Povey, G. ; Underwood, I. ; Haas, H.: Using a CMOS camera sensor for visible light communication. In: Globecom Workshops (GC Wkshps), 2012 IEEE, 2012, S. 1244–1248

[Deu96] Deutsch, P.: RFC 1951: DEFLATE Compressed Data Format Specification version 1.3. Mai 1996

[DGV03] Debaty, Philippe ; Goddi, Patrick ; Vorbau, Alex: Integrating the Physical World with the Web to Enable Context-Enhanced Services / Hewlett Packard Laboratories Palo Alto. 2003. – Forschungsbericht

[DH76] Diffie, Whitfield ; Hellman, Martin E.: New Directions in Cryptography. In: IEEE Transactions on Information Theory 22 (1976), Nr. 6

[Dig12] Digital Living Network Alliance: DLNA for Industry. Online: `http://www.dlna.org/dlna-for-industry` (Zuletzt geprüft: 29.09.2015), 2012

[DMA+10] Dixon, Colin ; Mahajan, Ratul ; Agarwal, Sharad ; Brush, A. J. ; Lee, Bongshin ; Saroiu, Stefan ; Bahl, Victor: The home needs an operating system (and an app store). In: Proceedings of the Ninth ACM SIGCOMM Workshop on Hot Topics in Networks. New York, NY, USA : ACM, 2010 (Hotnets '10). – ISBN 978–1–4503–0409–2

[DMA+12] Dixon, Colin ; Mahajan, Ratul ; Agarwal, Sharad ; Brush, A. J. ; Lee, Bongshin ; Saroiu, Stefan ; Bahl, Victor: An Opertaing System for the Home. In: Proceedings of the 9th USENIX Symposium on Networked Systems Design and Implementation, 2012 (NSDI '12)

[DOP11] Donohoo, B.K. ; Ohlsen, C. ; Pasricha, S.: AURA: An application and user interaction aware middleware framework for energy optimization in mobile devices. In: Computer Design (ICCD), 2011 IEEE 29th International Conference on, 2011, S. 168–174

[DR08] Dierks, T. ; Rescorla, E.: RFC 5246: The Transport Layer Security (TLS) Protocol Version 1.2. August 2008

[DTH06] Dhamija, Rachna ; Tygar, J. D. ; Hearst, Marti: Why Phishing Works. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems. New York, NY, USA : ACM, 2006 (CHI '06). – ISBN 1–59593–372–7, S. 581–590

[DY83] Dolev, Danny ; Yao, Andrew C.: On the security of public key protocols. In: Information Theory, IEEE Transactions on 29 (1983), Mar, Nr. 2, S. 198–208. – ISSN 0018–9448

[DZLT09] Dressler, E. ; Zender, R. ; Lucke, U. ; Tavangarian, D.: A multi-layer approach for cross-technology communication in a pervasive community. In: Mobile and Ubiquitous Systems: Networking Services, MobiQuitous, 2009. MobiQuitous '09. 6th Annual International, 2009, S. 1–6

[Ebe11] Eberhardt, Birgid (Hrsg.): AAL Anwendungsszenarien. VDE, 2011. – ISBN 978–3–925512–22–3

[Eck08] Eckert, Claudia: IT-Sicherheit, Konzepte - Verfahren - Protokolle. 5. Auflage. Oldenbourg Wissenschaftsverlag GmbH, 2008

[Ell03a] Ellison, C.: DeviceSecurity:1 Service Template. Online: http://www.upnp.org/standardizeddcps/documents/DeviceSecurity_1.0cc_001.pdf (Zuletzt geprüft: 29.09.2015), November 2003

[Ell03b] Ellison, C.: SecurityConsole:1 Service Template. Online: http://www.upnp.org/standardizeddcps/documents/SecurityConsole_1.0cc.pdf (Zuletzt geprüft: 29.09.2015), November 2003

[Ell03c] Ellison, C.: UPnP™Security Ceremonies Design Document. Online: http://www.upnp.org/download/standardizeddcps/UPnPSecurityCeremonies_1_0secure.pdf (Zuletzt geprüft: 29.09.2015), Oktober 2003

[EM14] Erven, Scott ; Merdinger, Shawn: Just what the doctor ordered? In: DEFCON 2014, 2014

[Eng02] Englander, Rob: Java and SOAP. O'Reily & Associates, 2002. – 35 S. – ISBN 978–0596001759

[ERA09] Eisenhauer, M. ; Rosengren, P. ; Antolin, P.: A Development Platform for Integrating Wireless Devices and Sensors into Ambient Intelligence Systems. In: Sensor, Mesh and Ad Hoc Communications and Networks Workshops, 2009. SECON Workshops '09. 6th Annual IEEE Communications Society Conference on, 2009

[Exc11] Exceptional Innovation, LLC: Life|ware. Online: http://www.life-ware.com/technology/dpws.php (Zuletzt geprüft: 29.09.2015), July 2011

[FIN11] FINDER S.P.A: 36 Series – Miniature PCB relays 10A – Datenblatt, 2011

[Fle15] Fleishman, Glenn: Das Internet der (verräterischen) Dinge | Technology Review. Online: http://heise.de/-2523589 (Zuletzt geprüft: 29.09.2015), 2015

[FMS01] Fluhrer, Scott ; Mantin, Itsik ; Shamir, Adi: Weaknesses in the Key Scheduling Algorithm of RC4. Version: 2001. http://dx.doi.org/10.1007/3-540-45537-X_1. In: Vaudenay, Serge (Hrsg.) ; Youssef, Amr (Hrsg.): Selected Areas in Cryptography Bd. 2259. Springer Berlin / Heidelberg, 2001, 1-24

[FS00] Ferguson, Niels ; Schneier, Bruce: A Cryptographic Evaluation of IPsec / Counterpane Internet Security, Inc. 2000. – Forschungsbericht

[FZ09] Finger, P. ; Zeppenfeld, K.: SOA und WebServices. Springer London, Limited, 2009 (Informatik im Fokus). – ISBN 9783540769910

[G+06] Goodrich, M.T. u. a.: Loud and clear: Human-verifiable authentication based on audio. In: 26th IEEE International Conference on Distributed Computing Systems IEEE, 2006

[GA14] Gailly, Jean loup ; Adler, Mark: zlib Home Site. Online: http://www.zlib.net/ (Zuletzt geprüft: 29.09.2015), 2014

[GHN+07] Goodner, Marc ; Hondo, Maryann ; Nadalin, Anthony ; McIntosh, Michael ; Schmidt, Don: Understanding WS-Federation – Version 1.0. Mai 2007

[Git13] GitHub, Inc.: rtyley/spongycastle. Online: https://github.com/rtyley/spongycastle (Zuletzt geprüft: 29.09.2015), 2013

[GMKK+13] Garcia-Morchon, O. ; Kumar, S. ; Keoh, S. ; Hummen, R. ; Struik, R.: Security Considerations in the IP-based Internet

of Things (draft-garcia-core-security-06). Version: 2013. http://tools.ietf.org/html/draft-garcia-core-security-06

[HBE+07] Hoffmann, Mario ; Badii, Atta ; Engberg, Stephan ; Nair, Renjith ; Thiemert, Daniel ; Matthess, Manuel: Security, Trust and Privacy supported by Context-Aware Middleware. In: 18th WWRF Meeting - Multimedia Goes Mobile, 2007

[Hem06] Hemel, A.: Universal Plug and Play: Dead simple or simply deadly? Online: http://www.upnp-hacks.org/sane2006-paper.pdf (Zuletzt geprüft: 29.09.2015), April 2006

[HGMH+11] Heer, Tobias ; Garcia-Morchon, Oscar ; Hummen, René ; Keoh, SyeLoong ; Kumar, SandeepS. ; Wehrle, Klaus: Security Challenges in the IP-based Internet of Things. In: Wireless Personal Communications 61 (2011), Nr. 3, 527-542. http://dx.doi.org/10.1007/s11277-011-0385-5. – DOI 10.1007/s11277–011–0385–5. – ISSN 0929–6212

[HHA+10] Handte, Marcus ; Haroon, Muhammad ; Apolinarski, Wolfgang ; Rak, Zoltan ; Zambrano, Alberto ; Selvarajah, Kirusnapillai: D4.1 Secure Middleware Specification - Version 1.4 / Peces - Pervasive computing in embedded systems. 2010. – Forschungsbericht

[Hig11] Higgins, Kelly J.: Digital Certificate Authority Hacked, Dozens Of Phony Digital Certificates Issued. Online: http://ubm.io/1FedJsL (Zuletzt geprüft: 29.09.2015), 2011

[HIMB05] Henricksen, Karen ; Indulska, Jadwiga ; McFadden, Ted ; Balasubramaniam, Sasitharan: Middleware for Distributed Context-Aware Systems. In: Meersman, Robert (Hrsg.) ; Tari, Zahir (Hrsg.): On the Move to Meaningful Internet Systems 2005: CoopIS, DOA, and ODBASE Bd. 3760. Springer Berlin / Heidelberg, 2005. – ISBN 978–3–540–29736–9, S. 846–863

[HJJ+07] Hore, B. ; Jafarpour, H. ; Jain, R. ; Ji, Shengyue ; Massaguer, D. ; Mehrotra, S. ; Venkatasubramanian, N. ;

Westermann, U.: Design and Implementation of a Middleware for Sentient Spaces. In: Intelligence and Security Informatics, 2007 IEEE, 2007, S. 137–144

[HKF15] Hu, Vincent C. ; Kuhn, D. R. ; Ferraiolo, David F.: Attribute-Based Access Control. In: IEEE Computer (2015), Nr. 02.15. – ISSN 0018–9162

[HLP+09] Hernández, V. ; López, L. ; Prieto, O. ; Martínez, J.-F. ; García, A.-B. ; Da-Silva, A.: Security Framework for DPWS Compliant Devices. In: Third International Conference on Emerging Security Information, Systems and Technologies (2009)

[Ho12] Ho, Jin-Meng: A Versatile Suite of Strong Authenticated Key Agreement Protocols for Body Area Networks. In: 8th International Conference on Wireless Communication and Mobile Computing (IWCMC'12), IEEE, 2012, S. 683–688

[Hol04] Hollenbeck, S.: RFC 3749: Transport Layer Security Protocol Compression Methods. Mai 2004

[IDS02] Imamura, Takeshi ; Dillaway, Blair ; Simon, Ed: XML Encryption Processing and Syntax. 2002

[IEE04] IEEE Standards: 802.1X™-2004 – IEEE Standard for Local and metropolitan are networks – Port-Based Network Access Control. 2004

[IEE06] IEEE: IEEE 802.15.4-2006: IEEE Standard for Information technology – Telecommunciations and information exchange between systems – Local and metropolitan area networks – Specific Requirements, Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs) / Institute of Electrical and Electronics Engineers. 2006. – Forschungsbericht

[IEE07] IEEE: IEEE 802.11-2007: IEEE Standard for Information technology – Telecommunciations and information exchange between systems – Local and metropolitan area networks – Specific Requirements, Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications / Institute of Electrical and Electronics Engineers. 2007. – Forschungsbericht

[Int04] International Organisation for Standardization (ISO): ISO/IEC 21000-5:2004 - Information technology – Multimedia framework (MPEG-21) – Part 5: Rights Expression Language. April 2004

[IST04] IST Amigo Project: Ambient Intelligence for the networked home environment (Projektbeschreibung). September 2004

[Kap15] Kaps, Reiko: Google deckt erneut Missbrauch im SSL-Zertifizierungssystem auf | heise online. Online: `http://heise.de/-2583414` (Zuletzt geprüft: 29.09.2015), 2015

[KBC97] Krawczyk, H. ; Bellare, M. ; Canetti, R.: HMAC: Keyed-Hashing for Message Authentication (RFC 2104). 1997

[KED+12] Kyusakov, Rumen ; Eliasson, Jens ; Deventer, Jan van ; Delsing, Jerker ; Cragie, Robert: Emerging energy management standards and technologies – Challenges and application prospects. In: Emerging Technologies Factory Automation (ETFA), 2012 IEEE 17th Conference on, 2012. – ISSN 1946–0740, S. 1–8

[KFJ01] Kagal, Lalana ; Finin, Tim ; Joshi, Anupam: Trust-Based Security in Pervasive Computing Environments. In: Pervasive Computing 34 (2001), Dezember, Nr. 12, S. 154–157

[KKP14] Kurawar, Arwa ; Koul, Ayushi ; Patil, Viki T.: Survey of Bluetooth and Applications. In: International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) 3 (2014), August, S. 2832–2837. – ISSN 2278–1323

[KKU+15] Kasparick, Martin ; Konieczek, Björn ; Unger, Sebastian ; Golatowski, Frank ; Timmermann, Dirk: Making Advanced Telemedicine Affordable. In: Global Telemedicine and eHealth Updates: Knowledge Resources 8 (2015), S. 178–182. – ISSN 1998–5509

[KMS07] Kushalnagar, N. ; Montenegro, G. ; Schumacher, C.: IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals (RFC 4919). 2007

[Kra14] Kransen, Jeroen: jkransen/framboos · GitHub. Online: `https://github.com/jkransen/framboos` (Zuletzt geprüft: 29.09.2015), 2014

[KS07] Kupris, Gerald ; Sikora, Axel: ZigBee: Datenfunk mit IEEE 802.15.4 und ZigBee. Franzis Verlag GmbH, 2007 (Elektronik- & Elektrotechnik-Bibliothek). – ISBN 978–3–7723–4159–5

[KST+09] Kobsa, Alfred ; Sonawalla, Rahim ; Tsudik, Gene ; Uzun, Ersin ; Wang, Yang: Serial Hook-ups: A Comparative Usability Study of Secure Device Pairing Methods. In: Proceedings of the 5th Symposium on Usable Privacy and Security. New York, NY, USA : ACM, 2009 (SOUPS '09). – ISBN 978–1–60558–736–3, S. 10:1–10:12

[KSTU09] Kumar, Arun ; Saxena, Nitesh ; Tsudik, Gene ; Uzun, Ersin: A comparative study of secure device pairing methods. In: Pervasive and Mobile Computing 5 (2009), Nr. 6

[L+06] Lockhart, Hal u. a.: WS-Federation Version 1.1. Online: `http://public.dhe.ibm.com/software/dw/specs/ws-fed/WS-Federation-V1-1B.pdf` (Zuletzt geprüft: 29.09.2015), Dezember 2006

[Leg14] Legion of the Bouncy Castle Inc.: bouncycastle.org. Online: `https://www.bouncycastle.org/` (Zuletzt geprüft: 29.09.2015), 2014

[LSS07] Lee, Jin-Shyan ; Su, Yu-Wei ; Shen, Chung-Chou: A Comparative Study of Wireless Protocols: Bluetooth, UWB, ZigBee, and Wi-Fi. In: Industrial Electronics Society, 2007. IECON 2007. 33rd Annual Conference of the IEEE, 2007. – ISSN 1553–572X, S. 46–51

[M+05] McCune, J.M. u. a.: Seeing-is-believing: using camera phones for human-verifiable authentication. In: IEEE Symposium on Security and Privacy, 2005

[Mao04] Mao, Wenbo: A Structured Operational Modelling of the Dolev-Yao Threat Model. In: Security Protocols Bd. 2845. Springer Berlin Heidelberg, 2004. – ISBN 978–3–540–20830–3, S. 34–46

[Mas09] Masak, Dieter: Digitale Ökosysteme - Serviceorientierung bei dynamisch vernetzten Unternehmen. Springer, Berlin, 2009. – ISBN 3–540–79129–9

[MG07] Mayrhofer, Rene ; Gellersen, Hans: Shake Well Before Use: Authentication Based on Accelerometer Data. In: Pervasive Computing Bd. 4480. Springer Berlin Heidelberg, 2007. – ISBN 978–3–540–72036–2, S. 144–161

[Mic00] Microsoft Corporation: Understanding Universal Plug and Play – White Paper. Online: `http://www.upnp.org/download/UPNP_UnderstandingUPNP.doc` (Zuletzt geprüft: 29.09.2015), Juni 2000

[Mic06] Microsoft Corporation: Devices Profile for Webservices. Online: `http://specs.xmlsoap.org/ws/2006/02/devprof/devicesprofile.pdf` (Zuletzt geprüft: 29.09.2015), Februar 2006

[MJ14] Mavrogiannopoulos, Nikos ; Josefsson, Simon: GnuTLS. Online: `http://www.gnutls.org/` (Zuletzt geprüft: 29.09.2015), 2014

[MKGC09] Müller, Andreas ; Kinkelin, Holger ; Ghai, Sunil K. ; Carle, Georg: An assisted device registration and service access system

for future home networks. In: Wireless Days (WD), 2009 2nd IFIP, 2009, S. 5

[MKGC10] Müller, Andreas ; Kinkelin, Holger ; Ghai, Sunil K. ; Carle, Georg: A Secure Service Infrastructure for Interconnecting Future Home Networks Based on DPWS and XACML. In: Proceedings of the 2010 ACM SIGCOMM Workshop on Home Networks. New York, NY, USA : ACM, 2010 (HomeNets '10). – ISBN 978–1–4503–0198–5, S. 31–36

[MKHC07] Montenegro, G. ; Kushalnagar, N. ; Hui, J. ; Culler, D.: Transmission of IPv6 Packets over IEEE 802.15.4 Networks (RFC 4944). 2007

[MLH+08] Martínez, J.-F. ; López, M. ; Hernández, V. ; Jean-Marie, K. ; García, A.-B. ; López, L. ; Herrera, C. ; Sánchez-Alarcos, C.-J.: A security architectural approach for DPWS-based devices. In: CollECTeR Iberoamérica (2008)

[Mos05] Moses, Tim: eXtensible Access Control Markup Langugae (XACML) Version 2.0 (OASIS Spcification). 2005

[MPTG08] Moritz, G. ; Prüter, S. ; Timmermann, D. ; Golatowski, F.: Web services on deeply embedded devices with real-time processing. In: IEEE International Conference on Emerging Technologies and Factory Automation, 2008, S. 432–435

[MSSS09] Mulyadi, Indra H. ; Supriyanto, Eko ; Safri, Norlaili M. ; Satria, Muhammad H.: Wireless Medical Interface Using ZigBee and Bluetooth Technology. In: Modelling Simulation, 2009. AMS '09. Third Asia International Conference on, 2009, S. 276–281

[MZP+09] Moritz, G. ; Zeeb, E. ; Prüter, S. ; Golatowski, F. ; Timmermann, D. ; Stoll, R.: Devices Profile for Web Services in Wireless Sensor Networks: Adaptations and enhancements. In: IEEE Conference on Emerging Technologies Factory Automation, 2009

[Nat08] National Institute of Standards and Technology: The Keyed-Hash Message Authentication Code (HMAC) (FIPS PUB 198-1). 2008

[NMT+07] Naroska, Edwin ; Mevissen, Ron ; Tobies, Stephan ; Hanbidge, Rich ; Dimitrov, Todor: Amigo Training Document: Using Amigo Security Services – Version 2.0 / IST Amigo Project. 2007. – Forschungsbericht

[NPS08] Nissen, V. ; Petsch, M. ; Schorcht, H.: Service-orientierte Architekturen: Chancen und Herausforderungen bei der Flexibilisierung und Integration von Unternehmensprozessen. Gabler Verlag, 2008 (Gabler Edition Wissenschaft). – ISBN 9783835008151

[NS78] Needham, Roger M. ; Schroeder, Michael D.: Using Encryption for Authentication in Large Networks of Computers. In: Commun. ACM 21 (1978), Dezember, Nr. 12, S. 993–999. http://dx.doi.org/10.1145/359657.359659. – DOI 10.1145/359657.359659. – ISSN 0001–0782

[NYHR05] Neuman, C. ; Yu, T. ; Hartman, S. ; Raeburn, K.: RFC 4120: The Kerberos Network Authentication Service (V5). Juli 2005

[OAS04] OASIS: Web Services Security Rights Expression Language (REL) Token Profile. Online: http://docs.oasis-open.org/wss/oasis-wss-rel-token-profile-1.0.pdf (Zuletzt geprüft: 29.09.2015), Dezember 2004

[OAS06a] OASIS: Web Services Security Kerberos Token Profile 1.1. Online: http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-KerberosTokenProfile.pdf (Zuletzt geprüft: 29.09.2015), Februar 2006

[OAS06b] OASIS: Web Services Security: SAML Token Profile 1.1. Online: http://docs.oasis-open.org/wss/v1.1/

wss-v1.1-spec-os-SAMLTokenProfile.pdf (Zuletzt geprüft: 29.09.2015), Februar 2006

[OAS06c] OASIS: Web Services Security: SOAP Message Security 1.1. http://www.oasis-open.org/committees/download.php/16790/wss-v1.1-spec-os-SOAPMessageSecurity.pdf (Zuletzt geprüft: 29.09.2015), Februar 2006

[OAS06d] OASIS: Web Services Security SOAP Messages with Attachments (SwA) Profile 1.1. Online: http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-SwAProfile.pdf (Zuletzt geprüft: 29.09.2015), Februar 2006

[OAS06e] OASIS: Web Services Security UsernameToken Profile 1.1. Online: http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-UsernameTokenProfile.pdf (Zuletzt geprüft: 29.09.2015), Februar 2006

[OAS06f] OASIS: Web Services Security X.509 Certificate Token Profile 1.1. Online: http://docs.oasis-open.org/wss/v1.1/wss-v1.1-spec-os-x509TokenProfile.pdf (Zuletzt geprüft: 29.09.2015) : Februar, 2006

[OAS06g] OASIS: WS-Trust 1.3. Online: http://docs.oasis-open.org/ws-sx/ws-trust/200512/ws-trust-1.3-spec-cs-01.html (Zuletzt geprüft: 29.09.2015), November 2006

[OAS07a] OASIS: WS-SecureConversation 1.3. Online: http://docs.oasis-open.org/ws-sx/ws-secureconversation/200512/ws-secureconversation-1.3-os.html (Zuletzt geprüft: 29.09.2015), März 2007

[OAS07b] OASIS: WS-SecurityPolicy 1.2. Online: http://docs.oasis-open.org/ws-sx/ws-securitypolicy/200702/ws-securitypolicy-1.2-spec-os.pdf (Zuletzt geprüft: 29.09.2015), Juli 2007

[OAS09a] OASIS: Devices Profile for Web Services Version 1.1. Online: http://docs.oasis-open.org/ws-dd/dpws/1.1/os/

wsdd-dpws-1.1-spec-os.html (Zuletzt geprüft: 29.09.2015), Juli 2009

[OAS09b] OASIS: Web Services Dynamic Discovery (WS-Discovery) Version 1.1. Online: http://docs.oasis-open.org/ws-dd/discovery/1.1/os/wsdd-discovery-1.1-spec-os.html (Zuletzt geprüft: 29.09.2015), Juli 2009

[OAS14] OASIS: MQTT Version 3.1.1. Online: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html (Zuletzt geprüft: 29.09.2015), Oktober 2014

[OM09] Ohigashi, Toshihiro ; Morii, Masakatu: A Practical Message Falsification Attack on WPA. 2009

[Ope14] OpenSSL (openssl@openssl.org): OpenSSL: The Open Source toolkit for SSL/TLS. Online: https://www.openssl.org/ (Zuletzt geprüft: 29.09.2015), 2014

[Pö10] Pöhlsen, Stephan: Entwicklung einer Service-orientierten Architektur zur vernetzten Kommunikation zwischen medizinischen Geräten, Systemen und Applikationen, Universität zu Lübeck, Dissertation, 2010

[Pau12] Pauli, Darren: Hacked terminals capable of causing pacemaker deaths. Online: http://www.scmagazine.com.au/News/319508,hacked-terminals-capable-of-causing-pacemaker-deaths.aspx, Oktober 2012. – Zuletzt geprüft: 29.09.2015

[PB03] Petrie, Charles ; Bussler, Christoph: Service Agents and Virtual Enterprises: A Survey. In: IEEE Internet Computing 7 (2003), Juli, Nr. 4, S. 68–78. http://dx.doi.org/10.1109/MIC.2003.1215662. – DOI 10.1109/MIC.2003.1215662. – ISSN 1089–7801

[PLA08] González, José L. (Hrsg.) ; Plaza, Pierre (Hrsg.). PLASTIC Consortium: A B3G Service Platform: The IST PLASTIC Project / PLASTIC Consortium. 2008. – Forschungsbericht

[PT08] Pishva, Davar ; Takeda, Keiji: Product-based security model for smart home appliances. In: Aerospace and Electronic Systems Magazine, IEEE 23 (2008), Oct, Nr. 10, S. 32–41. http://dx.doi.org/10.1109/MAES.2008.4665323. – DOI 10.1109/MAES.2008.4665323. – ISSN 0885–8985

[Rö10] Röcker, Carsten: Services and Applications for Smart Office Environments- A Survey of State-of-the-Art Usage Scenarios. In: Proceedings of the International Conference on Computer and Information Technology , ICCIT 2010, 2010, S. 387–403

[Res15] Rescorla, Eric: The Transport Layer Security (TLS) Protocol Version 1.3 draft-ietf-tls-tls13-08. August 2015

[RHC+02] Román, Manuel ; Hess, Christopher ; Cerqueira, Renato ; Ranganathan, Anand ; Campbell, Roy H. ; Nahrstedt, Klara: Gaia: a middleware platform for active spaces. In: SIGMOBILE Mob. Comput. Commun. Rev. 6 (2002), Oktober, Nr. 4, 65–67. http://dx.doi.org/10.1145/643550.643558. – DOI 10.1145/643550.643558. – ISSN 1559–1662

[RHP+08] Ragouzis, Nick ; Hughes, John ; Philpott, Rob ; Maler, Eve ; Madsen, Pauls ; Scavo, Tom: Security Assertion Markup Language (SAML) V2.0 Technical Overview. 2008

[RJPS05] Röcker, Carsten ; Janse, Maddy D. ; Portolan, Nathalie ; Streitz, Norbert: User requirements for intelligent home environments: a scenario-driven approach and empirical cross-cultural study. In: Proceedings of the 2005 joint conference on Smart objects and ambient intelligence: innovative context-aware services: usages and technologies. New York, NY, USA : ACM, 2005 (sOc-EUSAI '05). – ISBN 1–59593–304–2, 111–116

[RM06] Rescorla, E. ; Modadugu, N.: RFC 4347: Datagram Transport Layer Security. April 2006

[RS 13] RS Components: Raspberry Pi Model B Datenblatt, 2013

[RSPS02] Raghunathan, V. ; Schurgers, C. ; Park, S. ; Srivastava, M.B.: Energy-aware wireless microsensor networks. In: Signal Processing Magazine, IEEE 19 (2002), Märzr, Nr. 2, S. 40–50. http://dx.doi.org/10.1109/79.985679. – DOI 10.1109/79.985679. – ISSN 1053–5888

[RWRS00] Rigney, C. ; Willens, S. ; Rubens, A. ; Simpson, W.: RFC 2138: Remote Authentication Dial In User Service (RADIUS). 2000

[RZL13] Roman, Rodrigo ; Zhou, Jianying ; Lopez, Javier: On the features and challenges of security and privacy in distributed internet of things. In: Computer Networks 57 (2013), Nr. 10, S. 2266 – 2279. http://dx.doi.org/10.1016/j.comnet.2012.12.018. – DOI 10.1016/j.comnet.2012.12.018. – ISSN 1389–1286

[SA04] Saint-Andre, P.: Extensible Messaging and Presence Protocol (XMPP): Core – RFC 3920 / Internet Engineering Task Force (IETF). 2004. – IETF RFC

[SB05] Saffiotti, A. ; Broxval, M.: PEIS Ecologies: Ambient Intelligence meets Autonomous Robotics. In: sOc-EUSAI conference on Smart Objects and Ambient Intelligence. Grenoble, France, Oktober 2005

[SB09] Shelby, Zach ; Bormann, Carsten: 6LoWPAN: The Wireless Embedded Internet. John Wiley & Sons, 2009 (Wiley Series on Communications Networking & Distributed Systems). – ISBN 978–0470747995

[SBC05] Sivaharan, T ; Blair, Gordon S. ; Coulson, G: GREEN: A Configurable and Re-Configurable Publish-Subscribe Middleware for Pervasive Computing. In: Building 3760 LNCS (2005), 732–749. http://eprints.lancs.ac.uk/12730/

[SBG+08] Saffiotti, A. ; Broxval, M. ; Gritti, M. ; LeBlanc, K. ; Lundh, R. ; Rashid, J. ; Seo, B.S. ; Cho, Y.J.: The PEIS-Ecology Project: vision and results. In: IEEE/RSJ Int. Conf. on

Intelligent Robots and Systems (IROS). Nice, France, September 2008

[Sch15] Scherschel, Fabian: Gefälschtes Microsoft-Zertifikat im Umlauf | heise online. Online: http://www.heise.de/newsticker/meldung/Gefaelschtes-Microsoft-Zertifikat-im-Umlauf-2576861.html (Zuletzt geprüft: 29.09.2015), 2015

[SEKA06] Saxena, N. ; Ekberg, J.-E. ; Kostiainen, K. ; Asokan, N.: Secure device pairing based on a visual channel. In: Security and Privacy, 2006 IEEE Symposium on, 2006, S. 6 pp.–313

[Sel06] Selén, Kristian: Upnp security in internet gateway devices. In: TKK T-110.5190 Seminar on Internetworking, 2006

[Sna05] Snader, Jon C.: VPNs Illustrated: Tunnels, VPNs, and IPsec. Addison Wesley Professional, 2005. – ISBN 0–321–14544–X

[SP08] Scarfone, Karen ; Padgette, John: Guide to Bluetooth Security (NIST Special Publication 800-121). 2008

[SS75] Saltzer, Jerome H. ; Schroeder, Michael D.: The protection of information in computer systems. In: Proceedings of the IEEE 63 (1975), Nr. 9, S. 1278–1308

[SSAC10] Stammberger, Kurt ; Semp, Monique ; Anand, M.B. ; Culler, David: Introduction to Security for Smart Object Networks. 2010

[SSP08] Swoboda, Joachim ; Spitz, Stephan ; Pramateftakis, Michael: Kryptographie und IT-Sicherheit. 1. Vieweg + Teubner Verlag, 2008 (IT-Sicherheit und Datenschutz)

[STU08] Soriente, Claudio ; Tsudik, Gene ; Uzun, Ersin: HAPADEP: Human-Assisted Pure Audio Device Pairing. In: Information Security Bd. 5222. Springer Berlin Heidelberg, 2008. – ISBN 978–3–540–85884–3, S. 385–400

[SW04] Sastry, Naveen ; Wagner, David: Security considerations for IEEE 802.15.4 networks. In: In WiSe '04: Proceedings of the 2004 ACM workshop on Wireless security, ACM Press, 2004, S. 32–42

[SW10] Saeed, A. ; Waheed, T.: An extensive survey of context-aware middleware architectures. In: Electro/Information Technology (EIT), 2010 IEEE International Conference on, 2010. – ISSN 2154–0357

[TB09] Tews, Erik ; Beck, Martin: Practical attacks against WEP and WPA. In: Proceedings of the second ACM conference on Wireless network security. New York, NY, USA : ACM, 2009 (WiSec '09). – ISBN 978–1–60558–460–7, S. 79–86

[TSP02] Tandler, P. ; Streitz, N. ; Prante, T.: Roomware-moving toward ubiquitous computers. In: Micro, IEEE 22 (2002), Nov, Nr. 6, S. 36–47. http://dx.doi.org/10.1109/MM.2002.1134342. – DOI 10.1109/MM.2002.1134342. – ISSN 0272–1732

[Unb13] Unbekannt: Internet Census 2012. Online: http://internetcensus2012.bitbucket.org/paper.html (Zuletzt geprüft: 29.09.2015), 2013

[Ung15] Unger, Sebastian: Sebastian Unger / WS4D Mobile Authenticator | GitLab. Online: https://gitlab.amd.e-technik.uni-rostock.de/sebastian.unger/ws4d-mobile-authenticator/wikis/home (Zuletzt geprüft: 29.09.2015), September 2015

[UPn08] UPnP Forum: UPnP Device Architecture 1.1. Online: http://www.upnp.org/specs/arch/UPnP-arch-DeviceArchitecture-v1.1.pdf (Zuletzt geprüft: 29.09.2015), Oktober 2008

[UPT12a] Unger, S. ; Pfeiffer, S. ; Timmermann, D.: Dethroning Transport Layer Security in the Embedded World. In:

New Technologies, Mobility and Security (NTMS), 2012 5th International Conference on, 2012. – ISSN 2157–4952, S. 1–5

[UPT12b] Unger, Sebastian ; Pfeiffer, Stefan ; Timmermann, Dirk: How much security for switching a light bulb – The SOA way. In: Wireless Communications and Mobile Computing Conference (IWCMC), 2012 8th International, 2012, S. 1034–1039

[UT14] Unger, S. ; Timmermann, D.: Bridging the UI gap for authentication in smart environments. In: Computers and Communication (ISCC), 2014 IEEE Symposium on, 2014, S. 1–6

[UT15] Unger, Sebastian ; Timmermann, Dirk: DPWSec: Devices Profile for Web Services Security. In: 2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing: EWSN-AP 2015 (IEEE ISSNIP 2015 - EWSN-AP 2015). Singapore, Singapore, April 2015

[UZG+10] Unger, Sebastian ; Zeeb, Elmar ; Golatowski, Frank ; Grandy, Holger ; Timmermann, Dirk: Extending the Devices Profile for Web Services for Secure Mobile Device Communication. In: Internet of Things Conference - WS4: The 4th International Workshop on Trustworthy Internet of People, Things & Services, 2010

[van14] van Engelen, Robert A.: gSOAP: SOAP C++ Web Services. Online: `http://www.cs.fsu.edu/~engelen/soap.html` (Zuletzt geprüft: 29.09.2015), 2014

[vSS10] van Dijk, Hylke W. ; Schwartz, Ramon S. ; Scholten, Hans: Middleware for Deterministic Dynamically Reconfigurable Networked Embedded Systems - D3.3: Non-Functinal Service Composition Model for Security / Artemis. 2010. – Forschungsbericht

[W3C00] W3C: World Wide Web Consortium: SOAP Messages with Attachments. `http://www.w3.org/TR/SOAP-attachments` (Zuletzt geprüft: 29.09.2015), Dezember 2000

[W3C06a] W3C: World Wide Web Consortium: Web Services Policy 1.2 - Attachment (WS-PolicyAttachment). Online: http://www.w3.org/Submission/WS-PolicyAttachment/ (Zuletzt geprüft: 29.09.2015), April 2006

[W3C06b] W3C: World Wide Web Consortium: Web Services Eventing (WS-Eventing). Online: http://www.w3.org/Submission/2006/SUBM-WS-Eventing-20060315/ (Zuletzt geprüft: 29.09.2015), März 2006

[W3C07] W3C: World Wide Web Consortium: Web Services Policy 1.5 - Framework. Online: http://www.w3.org/TR/ws-policy/ (Zuletzt geprüft: 29.09.2015), September 2007

[Wei91] Weiser, Mark: The Computer for the 21st Century. In: Scientific American 265 (1991), Nr. 3, S. 94–104. http://dx.doi.org/doi:10.1038/scientificamerican0991-94. – DOI doi:10.1038/scientificamerican0991–94. – ISSN 0036–8733

[Wen15] Wendt, Johannes: IT-Sicherheit: Auch Medizintechnik lässt sich hacken | ZEIT ONLINE. Online: http://www.zeit.de/digital/internet/2015-04/medizintechnik-krankenhaus-it-sicherheit (Zuletzt geprüft: 29.09.2015), 2015

[WI13] Wu, Chwan-Hwa ; Irwin, J. D.: Introduction to Computer Networks and Cybersecurity. 1. CRC Press, 2013. – ISBN 978–1–4665–7213–3

[XCS+06] Xiao, Yang ; Chen, Hsiao-Hwa ; Sun, Bo ; Wang, Ruhai ; Sethi, Sakshi: MAC security and security overhead analysis in the IEEE 802.15.4 wireless sensor networks. In: EURASIP Journal on Wireless Communications and Networking 2006 (2006), April, S. 81–92. http://dx.doi.org/10.1155/WCN/2006/93830. – DOI 10.1155/WCN/2006/93830. – ISSN 1687–1472

[ZDLT08] Zender, R. ; Dressler, E. ; Lucke, U. ; Tavangarian, D.: Meta-Service Organization for a Pervasive University. In:

Pervasive Computing and Communications, 2008. PerCom 2008. Sixth Annual IEEE International Conference on, 2008, S. 400–405

[ZDLT09] Zender, R. ; Dressler, E. ; Lucke, U. ; Tavangarian, D.: Multi-level interoperability for pervasive communication networks. In: Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on, 2009, S. 1–3

[ZLT10] Zender, R. ; Lucke, U. ; Tavangarian, D.: SOA Interoperability for Large-Scale Pervasive Environments. In: Advanced Information Networking and Applications Workshops (WAINA), 2010 IEEE 24th International Conference on, 2010, S. 545–550

[ZSIH09] Zhang, Weishan ; Schütte, Julian ; Ingstrup, Mads ; Hansen, Klaus: A Genetic Algorithms-Based Approach for Optimized Self-protection in a Pervasive Service Middleware. In: Baresi, Luciano (Hrsg.) ; Chi, Chi-Hung (Hrsg.) ; Suzuki, Jun (Hrsg.): Service-Oriented Computing Bd. 5900. Springer Berlin / Heidelberg, 2009. – ISBN 978–3–642–10382–7, S. 404–419

[ZSS11] Zhao, Ran ; Selvarajah, Kirusnapillai ; Speirs, Neil: Development tools for pervasive computing in embedded systems (PECES) middleware. In: Wireless Information Networks and Systems (WINSYS), 2011 Proceedings of the International Conference on, 2011, S. 21–26

Thesen

1. Technologien wie Smart Homes, Ambient Assisted Living und intelligente Umgebungen werden sich nicht etablieren, bis sie den existierenden Sicherheitsbedrohungen umfassend begegnen.

2. In intelligenten Umgebungen besteht ein hohes Maß an Protokollheterogenität auf allen Ebenen des TCP/IP-Ebenenmodells. Dazu gehören ebenfalls Technologien, wie Bluetooth oder ZigBee, die sich nicht in das Ebenenmodell einfügen.

3. Um dem vorherrschenden Maß an Protokollheterogenität zu begegnen, muss eine Sicherheitsinfrastruktur, die Protokollinteroperabilität auch auf Dienstebene anvisiert, auf Anwendungsebene implementiert sein.

4. Sicherheitsmechanismen lassen sich am effizientesten auf möglichst niedrigen Schichten umsetzen. Ein Konzept auf hoher Schicht zur Gewährleistung zusätzlicher Funktionalität darf keine dramatischen Performance-Einbußen bedeuten, da sonst die Akzeptanz gefährdet ist.

5. Sicherheitsmechanismen können für den Nutzer nicht völlig unsichtbar sein, müssen jedoch einfach und intuitiv nutzbar sein.

6. Sicherheitsmechanismen, die den laufenden Betrieb stören, werden vom Nutzer nach Möglichkeit deaktiviert, umgangen oder durch Vereinfachung geschwächt.

7. In den Netzwerken der anvisierten Szenarien wie Smart Homes, Ambient Assisted Living und intelligenten Umgebungen befinden sich einige wenige bis mehrere Multimediageräte, die sich dadurch auszeichnen, dass

sie sich im direkten Wirkbereich ihrer Nutzer befinden, z.T. auch dauerhaft in unmittelbarer Nähe. Dieser Geräte bieten oft eine hohe Anzahl verschiedener Nutzerschnittstellen.

8. Die Sicherheitsinfrastrukturkomponenten klassischer Rechnernetze sind nutzerorientiert. Eine Sicherheitsinfrastruktur für verteilte eingebettete Systeme bedarf geräteorientierter Konzepte.

9. Neben Multimediageräten befinden sich in den Netzwerken der anvisierter Szenarien viele Geräte mit starker Ressourceneinschränkung und einige wenige Geräte, die quasi keiner Beschränkung unterliegen. Das bedeutet, dass sie regelrecht über unbegrenzte Ressourcen wie Taktfrequenz, Speicher, Bandbreite und Stromversorgung verfügen. Wenige starke Teilnehmer können viele schwache entlasten.

10. Die Verwendung und Adaption einer weit verbreiteten und zeitlich bewährten Grundtechnologie wie der Web Service Security Spezifikationssuite begünstigt die Akzeptanz und die Interoperabilität der entwickelten Lösung DPWSec.

11. Die Bewahrung der architektonischen Trennung von DPWS-Devices und -Clients führt zu kompakteren Bibliotheken und Implementierungen für die schwachen Teilnehmer.

12. Auf einen Teil der Zustandslosigkeit bezüglich authentifizierter Entitäten zu verzichten, ist nicht nur akzeptabel, sondern auch wünschenswert.

13. Das Request-Response-Kommunikationsmuster ist das heute gängigste.

14. Dienstinteroperabilität ist in den anvisierten Szenarien möglich.

15. Um den umfassenden Sicherheitsbedrohungen zu begegnen, darf nicht der Frage nachgegangen werden, welches Maß an Sicherheit auf einer bestimmten Plattform möglich ist. Stattdessen muss eine Lösung die Heterogenität der Leistungen berücksichtigen. Anschließend kann untersucht werden, welche Rechenleistung notwendig ist, um diese Lösung ausführen zu können.

Abstract

Internet of things, smart homes, ambient assisted living — they all are supposed to enhance and simplify our daily lives. However, they also have the potential to threaten our privacy or even life and limb. Infrastructures affecting every aspect of our existence need to be thoroughly protected.

At this point, it comes to coping with several challenges at once. Countless embedded systems with severe resource constraints need to be protected while they cannot run widely-deployed, well-known security infrastructures we designed to protect classical computer networks. Protocol heterogeneity is the same of a severe issue as the fact, that these systems are located directly at their users who must be capable of securely operating them – even as technical non-experts.

This work proposes DPWSec, a security infrastructure for distributed embedded system based on the Devices Profile for Web Services (DPWS). It will be shown that DPWSec comprises a comprehensive security infrastructure, as it fulfills a thoroughly compiled requirements list. Towards its end, this work shows, that the principles of DPWSec can even be ported to different base technologies, which provides the desired secure protocol interoperability.

Keywords — Ambient Assisted Living, ambient intelligence, communication security for embedded devices, DPWS, DPWSec, intelligent environments, internet of things

Kurzzusammenfassung

Visionen wie das Internet der Dinge, Smart Homes oder das Ambient Assisted Living sollen uns unser tägliches Leben vereinfachen und uns unterstützen. Sie bergen jedoch auch reale Gefahren für unsere Privatsphäre oder gar für Leib und Leben. Infrastrukturen, die so eng mit allen Aspekten unseres Lebens verflochten sind, bedürfen bedingungslosen Schutz.

Dabei gilt es, verschiedene Herausforderungen zu meistern. So haben all die genannten Visionen gemeinsam, dass in ihnen zahllose Kleinstgeräte ihren Dienst verrichten, die nicht über die Kapazitäten verfügen, die bekannten Sicherheitsinfrastrukturen klassischer Rechnernetze auszuführen. Daneben befinden sie sich in unmittelbarer Nähe zu ihren Nutzern und müssen auch von technischen Laien sicher betrieben werden können. Die vorhandene Protokollheterogenität in ambienten eingebetteten Systemen stellt ebenfalls besondere Herausforderungen.

Diese Arbeit schlägt DPWSec vor, eine Sicherheitsinfrastruktur für verteilte eingebettete Systeme auf Basis des Geräteprofils für Web Services, DPWS. Es wird gezeigt, dass DPWSec eine umfassende Sicherheitsinfrastruktur bereitstellen kann, die einer ausführlichen Anforderungsliste genügt. Im späteren Verlauf der Arbeit wird gezeigt, dass sich die Kernkonzepte von DPWSec auf alternative Basistechnologien übertragen lassen, wodurch eine sichere Protokollinteroperabilität ermöglicht wird.

Schlagwörter — Ambient Assisted Living, ambiente Intelligenz, DPWS, DPWSec, intelligente Umgebungen, Internet der Dinge, Kommunikationssicherheit für eingebettete Systeme

Eigene Publikationen

[KKU+15] Martin Kasparick; Björn Konieczek; **Sebastian Unger**; Frank Golatowski; Dirk Timmermann: Making Advanced Telemedicine Affordable. In Global Telemedicine and eHealth Updates: Knowledge Resources, Vol. 8, pp. 178-182, ISSN:1998-5509, Luxemburg, Luxemburg, April 2015

[UT15] **Sebastian Unger**; Dirk Timmermann: DPWSec: Devices Profile for Web Services Security. In 2015 IEEE Tenth International Conference on Intelligent Sensors, Sensor Networks and Information Processing: EWSN-AP 2015, ISBN: 978-1-4799-8055-0, Singapur, Singapur, April 2015

[UT14] **Sebastian Unger**; Dirk Timmermann: Bridging the Gap for Authentication in Smart Environments. In 19th IEEE Symposium on Computers and Communications (IEEE ISCC 2014), Funchal, Portugal, Juni 2014

[Ung14] **Sebastian Unger**: Liest die NSA Patientendaten? (Eingeladener Vortrag). In 9. Symposium Perioperative Medizin des Klinikums Südstadt Rostock, Rostock, Deutschland, März 2014

[ELW+13] Eichelberg, M.; Lipprandt, M.; Welge, R.; Busch, B.; Kabitzsch, K.; Laurila-Duersch, J.; Heusinger, S.; Golatowski, F.; **Unger, S.**; Moritz, G.; Gök, M.; Bothe, M.; Eberhardt, B.; Gaugler, J.; Lux, W.: Leitfaden interoperable Assistenzsysteme - vom Szenario zur Anforderung Teil 2 der Publikationsreihe „Interoperabilität von AAL-Systemkomponenten“ - DKE-Arbeitskreis STD 1811.0.12 „AAL-Interoperabilität“, VDE-Verlag, ISBN: 978-3-8007-3515-0, Berlin , Deutschland, Juni 2013

[Ung13] **Sebastian Unger**: Secure Web Services für ambiente eingebettete Systeme. In Interdisziplinäres Symposium Zu viel des Smarten? Wie viel Assistenz braucht der Mensch?, Rostock, Deutschland, Juni 2013

[Ung12] **Sebastian Unger**: A comprehensive Security Framework for Distributed Systems of Resource-constrained Devices. In 1st WS4D Workshop, Rostock, Deutschland, November 2012

[UPT12b] **Sebastian Unger**; Stefan Pfeiffer; Dirk Timmermann: How much Security for Switching a Light Bulb - The SOA Way. In 8th International Conference on Wireless Communications and Mobile Computing, pp. 1034-1039, ISBN: 978-1-4577-1379-8, Limassol, Zypern, August 2012

[UPT12a] **Sebastian Unger**; Stefan Pfeiffer; Dirk Timmermann: Dethroning Transport Layer Security in the Embedded World. In The Fifth IFIP International Conference on New Technologies, Mobility and Security, keine Seitenzahlen, ISBN: 978-1-4673-0229-6, Istanbul, Türkei, Mai 2012

[PLUT12] Stefan Pfeiffer; Andreas Lehmann; **Sebastian Unger**; Dirk Timmermann: Secure Information Flow Awareness for Smart Wireless eHealth Systems. In 9th International Multi-Conference on Systems, Signals and Devices (SSD'12), Chemnitz, Deutschland, März 2012

[UM12] **Sebastian Unger**; Guido Moritz: A comprehensive Security Framework for Distributed Systems of Resource-constrained Devices. Position Paper für IETF-Workshop on Smart Object Security, Paris, Frankreich, März 2012

[UZG+10] **Sebastian Unger**; Elmar Zeeb; Holger Grandy; Frank Golatowski; Dirk Timmermann: Extending the Devices Profile for Web Services for Secure Mobile Device Communication. In 4th International Workshop on Trustworthy Internet of People, Things & Services colocated at the Internet of Things 2010, ISBN: 978-1-4244-7414-1, Tokyo, Japan, November 2010

Betreute studentische Arbeiten

[Stud1] Robert Balla; Hannes Domschke: Analyse der Performance verschiedener Kryptoalgorithmen in verschiedenen Programmiersprachen und auf verschiedenen Architekturen. *Seminararbeit 2014*

[Stud2] Hannes Domschke: Implementierung eines Proof-of-Concept zur indirekten Authentifizierung in einem Smart-Home-Szenario. *Bachelorarbeit 2013*

[Stud3] Hannes Domschke: Erstellung eines Überblicks über Authentifizierungsmethoden in existierenden Middlewaretechnologien und bestehenden Szenarien. *Literaturarbeit 2013*

[Stud4] Arne Wall: Analyse und Klassifikation von Sicherheitsmechanismen in Middlewares für intelligente Umgebungen. *Literaturarbeit 2013*

[Stud5] Martin Sonnberg: Sicherheitmechanismen in Technologien zur spontanen Gerätevernetzung. *Vortragsseminar 2013*

Eidesstattliche Erklärung

Ich erkläre hier an Eides statt, dass ich die vorliegende Arbeit selbstständig und nur unter Vorlage der angegebenen Literatur und Hilfsmittel angefertigt habe.

Rostock, den 26. März 2016

.....

Sebastian Unger